AN ARGUS SPECIALIST PUBLICATION

FEBRUARY 1986

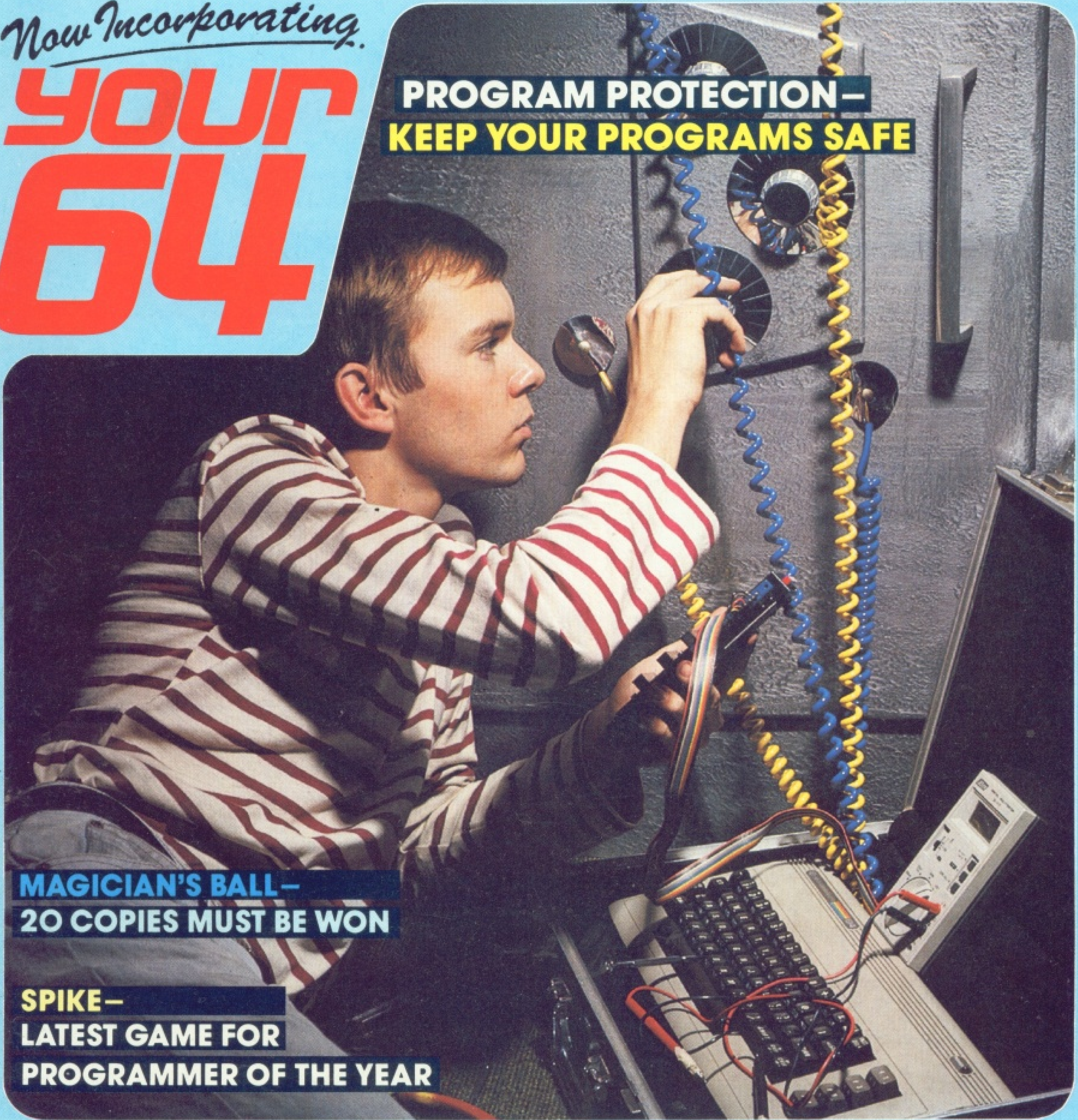
£1.00

# Your COMMODORE

## YOUR BEST INDEPENDENT COMMODORE MAGAZINE

Now Incorporating

## YOUR 64



PROGRAM PROTECTION—
KEEP YOUR PROGRAMS SAFE

MAGICIAN'S BALL—
20 COPIES MUST BE WON

SPIKE—
LATEST GAME FOR
PROGRAMMER OF THE YEAR

FEBRUARY 1986

VOLUME 2
NUMBER 5

Editor:
Stuart Cooke

Assistant Editor:
Marie Curry

Advertisement Manager:
Mike Segrue

Advertisement Copy Control:
Laura Champion

Group Editor:
Dave Bradshaw

Group Managing Editor:
Wendy Palmer

Managing Director:
Peter Welham

Origination:
Ebony Typesetting

Design:
Argus Design

Editorial & Advertisement Office
No 1 Golden Square,
London W1R 3AB.
Telephone: 01-437 0626
Telex: 8811896

Your Commodore is a monthly magazine appearing on the first Friday of each month.

Distribution by: Argus Press Sales & Distribution Ltd. 12-18 Paul Street, London EC2A 4JS. Printed by: Alabaster Passmore & Sons Ltd, Tovil, Maidstone, Kent.

Subscription rates upon application to Your Commodore Subscriptions Department, Infonet Ltd, Times House, 179 The Marlowes, Hemel Hempstead, Herts. HP1 1BB.



# CONTENTS

## FEATURES



## SERIES



## REGULARS



## COMPETITIONS



## GAMES AND UTILITIES

# DATA STATEMENTS

## Soft in the Head –

YES, IT HAD TO HAPPEN SOME TIME. Superman has got fed up of doing his quick change act in the confines of a phone box and will now be executing this incredible feat of contortion inside your computer. Beyond has recently launched *Superman The Computer Game*, which features Superman, on the side of good, fighting Darkseid for control of a metropolis.

According to Beyond, the game contains a minimum of rules and is punctuated with breathtaking animated sequences. It's not a bird or a plane but it is £9.95.

Also in Beyond's autumn release package were EnigmaForce, the sequel to Shadowfire and Spy vs Spy: The Island Caper, sequel to Spy vs Spy (as if you hadn't guessed). Both cost £9.95 on cassette.

Another superman, international goalkeeper Ray Clemence, has put his stamp of approval on Macmillan Software's new release *World Cup Soccer*. The package contains two programs and a book which provide information on the skills techniques and secrets of some of the world's top players. Goalie Ray commented: "*World Cup Soccer* is a must for any serious and dedicated soccer fan. Once you've mastered this you can sit back and watch the 1986 World Cup through the eyes of a real professional."



**Goalie Ray with the team**

**Five for under a tenner**






Quicksilva has come up with an arcade strategy game for the 64 called *Deathwake*, which features you as a brilliant admiral trying to rebuild your shattered forces so that you can destroy the enemy's special research lab, hidden in the depths of a mountainside. If you don't, they will finish developing the Ultimate Weapon – an atomic bomb. (Hasn't some one already developed it?) If you want to gain lost territory from the evil Alliance then you'll have to fork out £7.95 for the privilege.

Ultimate has two new titles for the C64; *Dragon Skulle* and *Outlaws* – shouldn't that be Autlors? They both cost £9.95 on the C64 and are embellished with Ultimate's unmistakable brand of art work.

Hewson Consultants want you to let the train take the strain and buy *Southern Belle* for the C64. This steam locomotive simulator is said by its makers to be for the more 'sophisticated' game player and it is rumoured that 'railway enthusiasts have even been buying computers specifically to sample its delight.' Holy smoke!

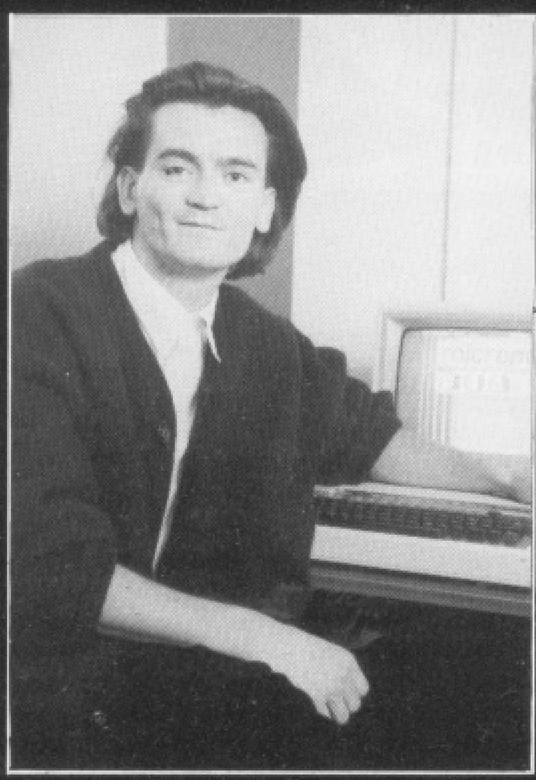
On to more serious software, and Impex has produced a program called *Font Factory* which is aimed at improving the output from a dot matrix printer. It reads any standard Commodore ASCII file, automatically formats and prints it. And you get a choice of eight different typefaces. It incorporates control of line width and spacing and justification. Also on the disk is a program called *Sign Writer* which allows you to produce banners using letters a foot high. You get both programs for £19.95.

Impex has also released *Fantastic Filer* and *Screen Dumper 64*. Both these programs are £12.95 each and available on disk only.

**Feargal and friend Casey Jones on computer by Hewson**

Domark, meanwhile, has decided to try and scare us all by launching *Friday the 13th, The Computer Game*, featuring a mad fiend called Jason who wanders round a holiday camp trying to get his homicidal way with all the innocent campers. Your job is, of course, to save them but mind you don't panic as this seems to infuriate the psychopath. £8.95 on cassette and £11.95 on disk, probably a game not to be played in the dark. The sadist's *Terrormolinos*!

Ariolasoft has launched a new range of software for the autumn which is unusual because the disk versions are under £10 – at £9.95. Frank Brunger, marketing and sales director, said: "The cost may be lower but the quality certainly isn't." The Cassette versions are £7.95 and the new titles are: *Axis Assassin, D-Bug, Bug Blitz, Kaiser* and *Saucer Attack*. All are available on the C64.

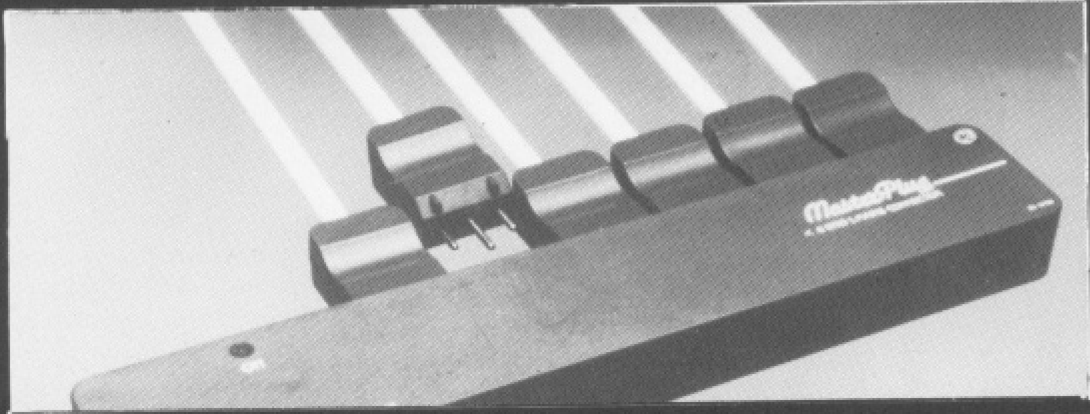


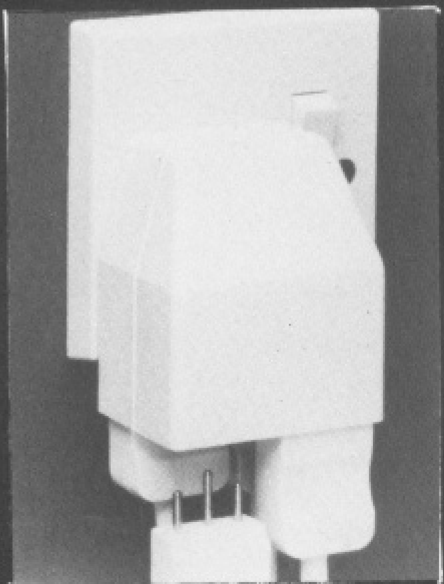
## In Touch

MICRONET HAS MOVED ITSELF INTO the glamorous world of pop music. Feargal Sharkey appeared on Celebrity Chatline and was greeted with an enormous response from Micronet members.

Feargal has himself been a member of Micronet for over a year and is very impressed with the service, saying: "Most of all I find it good relaxation." He also loves the Celebrity Chatline in particular. "I think it's very entertaining," he raved. "It's certainly a lot more entertaining than most of the programmes on TV at the moment. I'd rather watch Chatline than Coronation Street." Obviously an Eastenders fan.

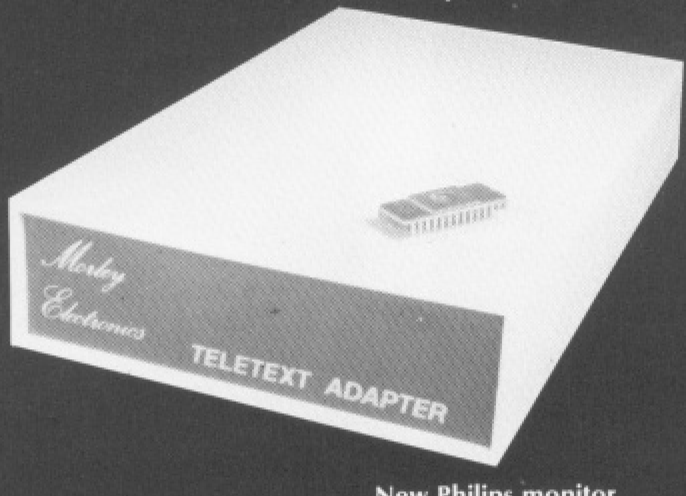
Micronet members have also been making an effort to help the survivors of the Mexican earthquake. In the first month of the Mexican Aid Appeal they raised over £180. Donations should be made payable to Mexican Aid and sent to the address below. Micronet members should call page *800119836



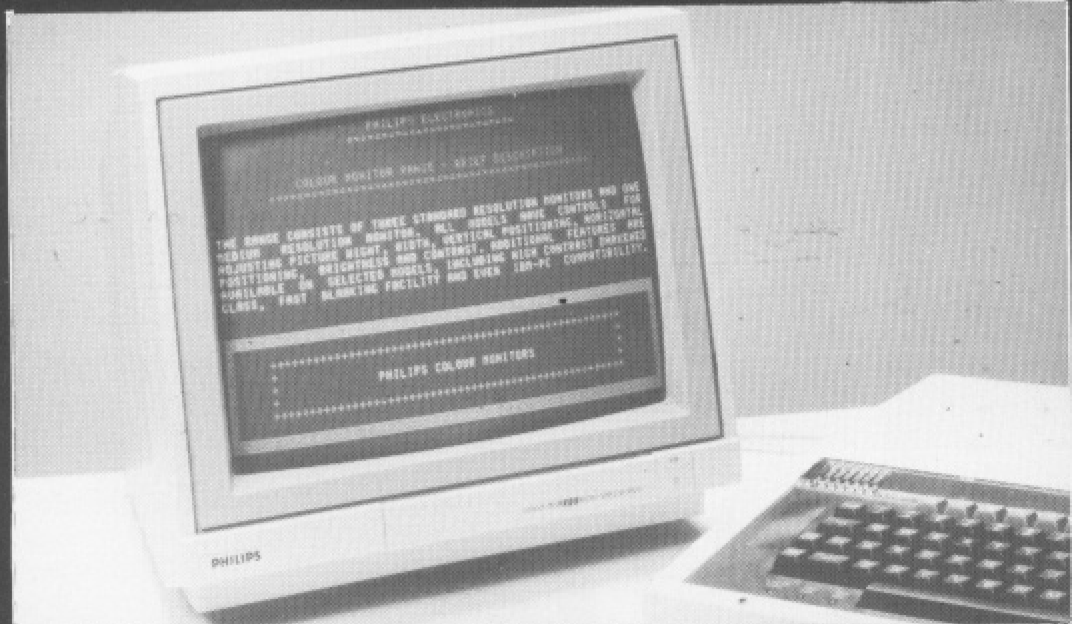


No more tangled wires

Teletext Adapter for the C64




New Philips monitor




## Hard Lines

MORLEY ELECTRONICS HAS NOW announced the availability of a Teletext Adapter for the C64. The C64 version of the Adapter plugs into the user expansion port and uses software to produce a simulated teletext display.

The Commodore version costs about £130 and anyone interested should contact Morley for more details.

There's also a new range of colour monitors now available from Philips. There are four models in the range and prices start at £220.

Philips policy is to improve the clarity, resolution and performance of monitors to keep in line with improvements made to home computers. Philips state that the monitors are designed to give superior quality and crispness for every computing need.

## Generally Speaking

FIREBIRD HAS NOW GOT A BABY SISTER – or should that be egg. British Telecom is forming a new software company, which will be totally separate from Firebird. The new outfit is to be called Rainbird and will be headed by its namesake Tony Rainbird.

First release from Rainbird is Island Logic's, *The Music System*. It has been available for the BBC computer for some time, but only now has a Commodore 64 version been perfected.

Rainbird will also be producing software for 16 bit computers including Commodore's latest little offering, the Amiga. Firebird will continue to burn brightly and independently producing games for 8 bit machines.

For those who are confused about the function of the new Data Protection Act 1984, the Data protection Registrar has produced a handy question and answer booklet to try and clarify the most important points. The Act is designed to protect individuals rights by allowing them to have access to personal information which various organisations may have on file. Subjects covered range from personal data held at schools and universities to registration for groups of companies.

If you're fed up with seeing those horrible tangled up wires around your computer then Conblock Electrical Limited may have come up with the solution to your problem. Now available are two new adaptors, one of which can take up to six plugs, the other up to four. Both are smaller and lighter than traditional ones and they certainly look a lot better, too.

Both come complete with plugs and are for use in any standard 13 amp socket. They also conform to the Electrical Equipment Safety Regulations 1985.

## Touch Line

**Beyond**, 3rd Floor, Lector Court, 151 Farringdon Rd, London EC1R 3AD, 01 837 2899
**Macmillan Software**, 4 Little Essex St, London WC2R 3LF, 01 836 6633
**Domark**, 01 947 5622
**Ariolasoft**, 8 Westminster Palace Gardens, Artillery Row, London SW1P 1RL
**Quicksilva**, Liberty Hse, 222 Regent St, London W1R 7DB, 01 439 0666
**Ultimate**, The Green, Ashby de la Zouch, Leics LE6 5JU, 0530 411485
**Hewson Consultants**, 56B Milton Trading Est, Milton, Abingdon, Oxon, 0235 832939
**Impex**, Metro Hse, Second Way, Wembley, Middx, HA9 0TY, 01 900 0999
**Micronet 800**, 8 Herbal Hill, London EC1R 5EJ, 01 278 3143
**Morely Electronics**, 1 Morley Place, Earsdon Rd, Shiremoor, Tyne & Wear, 091 2513883
**Philips**, Burston-Marsteller, 25 North Row, London W1R 2BY, 01 499 0414
**British Telecom**, Wellington Hse, Upper St Martins Lane, London EC1
**The Data Protection Registrar**, Springfield Hse, Water Kane, Wilmslow, Cheshire SK9 5AX, 0625 535777
**Conblock Electrical Ltd**, 1 Merridale Rd, Chapel Ash, Wolverhampton, W Midlands WV3 9RT, 0902 773737
**Mexican Aid**, 20 Holmes Rd, Kentish Town, London NW5 3AR or Account 03176762, Nat West Bank, Kentish Town Branch, London NW5 2DG

# COUNT ON YOUR COMMODORE

**In a mathematical special, Nick Hampshire shows you how to use the C64's arithmetic routines.**

## Numeric Variables, Types and Range

BASIC USES TWO DIFFERENT types of numbers, integer and floating point. An integer number is stored as two bytes giving a 16 bit signed number which can store numbers in the range +32767 to −32768. Floating point numbers require five bytes and can store much larger values in the range ± 1.70141183 E38 to ± 2.93873588 E −39. In the Basic interpreter all calculations, whether on integer or floating point values, are performed using the latter rather than simple integers or binary values. Consequently, all integer values are first converted to floating point format before any calculations are performed.

The format for the storage of an integer value is very simple, consisting of two bytes stored as low order/high order byte. Negative values are stored in a two's complement form, — the format is shown in Figure 1. Floating point values are stored in either packed form, occupying five bytes, or unpacked form in six bytes. Packed format is the normal mode for storing floating point variables in memory. Unpacked format is used when performing calculations upon floating point values. In either format there are three components of a floating point value, — the sign, the exponent and a four byte mantissa. In packed mode the sign is stored as bit seven of the most significant byte of the mantissa. In unpacked format the sign occupies its own byte.

## The Floating Point Accumulator

In order to perform arithmetic operations on any floating point value the interpreter needs temporary storage locations for the values being worked upon as well as the result. There are two principle work areas, they are known as 'floating point accumulator 1' and 'floating point accumulator 2'. These names are usually shortened to Fac 1 and Fac 2. Each floating accumulator occupies six bytes and Fac 1 starts at $61 while Fac 2 starts at $69. There are, in addition, three further areas where floating point numbers in packed format (occupying five bytes) are stored. These areas start at $57,$5C and $26. The format and location of the two floating accumulators is as follows:

| Locations Fac 1 | Fac 2 | Function |
|---|---|---|
| $61 | $69 | exponent + $80 |
| $62 | $6A | mantissa msb |
| $63 | $6B | mantissa byte 2 |
| $64 | $6C | mantissa byte 3 |
| $65 | $6D | mantissa lsb |
| $66 | $6E | sign ($FF = − and $00 = +) |

```
5 REM ** REAL NUMBER FORMAT (PACKED) **
10 A=0
20 C=PEEK(45)+PEEK(46)*256+2
30 INPUT" A REAL NUMBER";A
40 E=PEEK(C)
50 M1=PEEK(C+1)
60 M2=PEEK(C+2)
70 M3=PEEK(C+3)
80 M4=PEEK(C+4)
90 PRINT
100 PRINTE;M1;M2;M3;M4
105 IFE=0THENPRINT0:END
110 SG=SGN(64-(M1 AND 128))
120 N=(M1 AND127)+128
130 N=N*256+M2
140 N=N*256+M3
150 N=N*256+M4
160 N=N*2↑(E-160)*SG
200 PRINTN
```

**Program 1**

```
5 REM ** REAL NUMBER FORMAT (PACKED) **
10 A=0
20 C=PEEK(45)+PEEK(46)*256+2
30 INPUTB
35 IFB=0THENPRINT0;0;0;0;0:PRINT:GOTO230
40 EX=INT(LOG(ABS(B))/LOG(2))
50 E=EX+129
60 R=B-2↑EX
70 SG=SGN(-B)*64+64
80 T0=(R/2↑EX)*128
90 M1=INT(T0)+SG
100 T1=(T0-INT(T0))*256
110 M2=INT(T1)
120 T2=(T1-INT(T1))*256
130 M3=INT(T2)
140 T3=(T2-INT(T2))*256
150 M4=INT(T3)
160 PRINTE;M1;M2;M3;M4
170 PRINT
180 POKEC,E
190 POKEC+1,M1
200 POKEC+2,M2
210 POKEC+3,M3
220 POKEC+4,M4
230 PRINTA
```

**Program 2**

other locations used are:
$68 — overflow byte for Fac 1
$6F—sign comparison byte
$70—rounding byte for Fac 1

## How a Floating Point Number is Stored

The storage of a floating point number is fairly complex both in packed and unpacked format. The data used to store a floating point number can be divided into three components; the exponent, the sign and the mantissa. In the unpacked format, the exponent and sign both occupy one byte and the mantissa four bytes. The following is an explanation of each component of a floating point number.

**Exponent** — The exponent indicates the position of the decimal point within the number. Bit seven of the exponent byte indicates the sign of the exponent. Thus, if the exponent is positive, bit seven is set to one and, therefore, the value of the exponent byte will always be greater than 128. If the exponent is negative then bit seven is set to zero and the exponent value is less than 128. The exponent is stored as a power of two and is multiplied by the mantissa value to produce the final value. The following formulae can be used to convert a number N stored in the mantissa bytes (see paragraph on mantissa for calculation of N) into the full floating point number by multiplying it with a positive exponent:

Value = N * 2  (E-129)

To determine the exponent of a number, find the highest power of two which can be subtracted from the number. Thus, if the number is 18.256 then the highest power of two is 16 or 2↑4. The exponent value is positive, and therefore equals 129+4 or 133. The fact that the exponent is derived in this way means that the mantissa for two different values may be the same, with the difference being registered solely by the contents of the exponent. Thus, the floating point mantissa contents for the values 3.14159 (pi) and 6.28318 (pi*2) are identical:

3.14159 stored as — exponent 130 and mantissa 73,15,218,161
6.28318 stored as — exponent 131 and mantissa 73,15,218,161

As you can see, multiplying and dividing a floating point number by two is a very simple operation involving adding or subtracting one from the exponent. The range of the exponent is ± 2↑128. This equates approximately to ± 10 38.

**Sign** — The sign of the value is stored in unpacked — format as a single byte with a value of $FF for negative numbers for $00 for positive numbers. In packed format the sign is stored in bit seven of the highest byte of the mantissa. If bit seven is zero then the mantissa is positive, and if it is one then the mantissa is negative. Thus the unpacked floating point values for +2 and -2 are:

number +2 is — exponent 130 and mantissa 0,0,0,0
number -2 is — exponent 130 and mantissa 128,0,0,0

**Mantissa** — The mantissa is stored in four bytes less the most significant bit of the most significant byte of the mantissa which is used to store the sign bit. To convert a number stored in the mantissa into its numeric

```
033C              !CALCULATE (A+22)/(B*5)
033C              ! WHERE A AND B ARE INPUT FROM
033C              ! THE KEYBOARD.
033C              ! ENTRY AT SYS 49171.
033C              !
033C              ! RESULT IS PRINTED
033C              !
C000              *=$C000
C000 0000   AV             WOR 0
C002 0000   BV             WOR 0
C004 000000 TF1            BYT 0,0,0,0,0
C009 000000 TF2            BYT 0,0,0,0,0
C00E 000000 TF3            BYT 0,0,0,0,0
C013 A000   ENTRY          LDY #$00
C015 20CFFF L1             JSR $FFCF        !INPUT BYTE
C018 C90D                  CMP #$0D         !CARRIAGE RETURN?
C01A F006                  BEQ L2           !YES
C01C 990002                STA $0200,Y      !STORE BYTE
C01F C8                    INY              !DO NEXT
C020 D0F3                  BNE L1           !ALWAYS
C022 A900   L2             LDA #$00         !ZERO TERMINATOR
C024 990002                STA $0200,Y
C027 A900                  LDA #$00         !SET CHARGET TO
C029 857A                  STA $7A          !BUFFER
C02B A902                  LDA #$02
C02D 857B                  STA $7B
C02F 207900                JSR $0079
C032 208AAD                JSR $AD8A        !CONVERT TO # 0-65535
C035 20F7B7                JSR $B7F7        !MAKE INTEGER
C038 A514                  LDA $14          !STORE VALUE
C03A 8D00C0                STA AV           !IN TEMP
C03D A515                  LDA $15
C03F 8D01C0                STA AV+1
C042 A000   ENTRY1         LDY #$00
C044 20CFFF L3             JSR $FFCF        !INPUT BYTE
C047 C90D                  CMP #$0D         !CARRIAGE RETURN?
C049 F006                  BEQ L4           !YES
C04B 990002                STA $0200,Y      !STORE BYTE
C04E C8                    INY              !DO NEXT
C04F D0F3                  BNE L3           !ALWAYS
C051 A900   L4             LDA #$00         !ZERO TERMINATOR
C053 990002                STA $0200,Y
C056 A900                  LDA #$00         !SET CHARGET TO
C058 857A                  STA $7A          !BUFFER
C05A A902                  LDA #$02
C05C 857B                  STA $7B
C05E 207900                JSR $0079
C061 208AAD                JSR $AD8A        !CONVERT TO # 0-65535
C064 20F7B7                JSR $B7F7        !MAKE INTEGER
C067 A514                  LDA $14          !STORE VALUE
C069 8D02C0                STA BV           !IN TEMP
```

Program 3

equivalent use the following formulae:

N = 1+(M1 AND 127)+(M2+(M3 +M4/256)/256)/256)/128

where M1,M2,M3 and M4 are the mantissa bytes, with M1 the highest and M4 the lowest. When N has been obtained it should be multiplied by 2 (exponent — 129) to give the actual value. The program in Program 1 allows the input of a number, then prints the contents of the exponent and mantissa bytes for that number as it is stored in floating point. These values are then used by lines 90 to 120 to convert the floating point byte values back into the number.

To convert a number into floating point form is a slightly harder calculation and involves the following steps:
First find the highest power of two which can be subtracted from the number. E = the value of two to this highest power.
Secondly let R = the remainder after subtracting the value of 2 E.
The calculation is then as follows:
TO = (R/E)*128
M1 = INT(TO)+mantissa sign (sign =0 if positive 128 if negative)
T1 = (TO–INT(TO))*256
M2 = INT(T1)
T2 = (T1–INT(T1))*256
M3 = INT(T2)
T3 = (T2–INT(T2))*256
M4= INT(T3)
Where M1,M2,M3,M4 are the four mantissa byte values, M1 being the highest. The program in Program 2 does this conversion of a number input at the beginning of the program into the five bytes of a floating point format which are displayed on the screen. The program then checks by putting these values into the first variable in memory defined as a simple variable A in line 10.
The following are examples of the storage of some floating point numbers:

| Number | Exponent | M1 | M2 | M3 | M4 | Sign |
|---|---|---|---|---|---|---|
| 1 | $81 | $80 | $00 | $00 | $00 | $00 |
| –1 | $81 | $80 | $00 | $00 | $00 | $FF |
| .5 | $80 | $80 | $00 | $00 | $00 | $00 |
| .25 | $7F | $80 | $00 | $00 | $00 | $00 |
| 1E38 | $FF | $96 | $76 | $99 | $52 | $00 |
| 1E–39 | $00 | $A0 | $00 | $00 | $00 | $00 |

```
C06C A515      LDA $15
C06E 8D03C0    STA BV+1
C071 AD01C0    LDA AV+1      !GET FIRST VALUE
C074 AC00C0    LDY AV
C077 2091B3    JSR $B391     !FLOAT IT
C07A A204      LDX #<TF1     !STORE IN TEMP FAC1
C07C A0C0      LDY #>TF1
C07E 20D4BB    JSR $BBD4
C081 A900      LDA #$00      !VALUE 22 ($16)
C083 A016      LDY #$16
C085 2091B3    JSR $B391     !FLOAT IT
C088 A904      LDA #<TF1     !POINT TO TEMP
C08A A0C0      LDY #>TF1     !FAC1
C08C 2067B8    JSR $B867     !ADD
C08F A204      LDX #<TF1     !STORE IN TEMP FAC1
C091 A0C0      LDY #>TF1
C093 20D4BB    JSR $BBD4
C096 AD03C0    LDA BV+1      !GET SECOND VALUE
C099 AC02C0    LDY BV
C09C 2091B3    JSR $B391     !FLOAT IT
C09F A209      LDX #<TF2     !STORE IN TEMP FAC2
C0A1 A0C0      LDY #>TF2
C0A3 20D4BB    JSR $BBD4
C0A6 A900      LDA #$00      !GET VALUE 5
C0A8 A005      LDY #$05
C0AA 2091B3    JSR $B391     !FLOAT IT
C0AD A909      LDA #<TF2     !POINT TO TEMP
C0AF A0C0      LDY #>TF2     !FAC2
C0B1 2028BA    JSR $BA28     !MULTIPLY
C0B4 A904      LDA #<TF1     !POINT TO TEMP
C0B6 A0C0      LDY #>TF1     !FAC1
C0B8 200FBB    JSR $BB0F     !DIVIDE
C0BB A20E      LDX #<TF3     !STORE RESULT IN
C0BD A0C0      LDY #>TF3     !TEMP FAC3
C0BF 20D4BB    JSR $BBD4
C0C2 20DDBD    JSR $BDDD     !CONVERT TO STRING
C0C5 201EAB    JSR $AB1E     !PRINT STRING
C0C8 4C74A4    JMP $A474     !'READY.'
```

Table 1

## Using the Arithmetic Routines in a Machine Code Program

Using the arithmetic routines within the Basic interpreter can save the programmer a lot of time in program development. It can also considerably reduce the size of a machine code program. The only penalty is that in the program using eight or 16 bit values the interpreter routines will have a considerably slower run time than specially written routines. When faced with the necessity of having to use arithmetic routines the best procedure is to always use the interpreter routines and only replace these if the program is running too slow. A list of the main arithmetic routines within the C64 is shown in Table 1.

It is quite simple to utilise the interpreter arithmetic routines within a machine code program. The essential thing to remember is that the interpreter does all its calculations on floating point numbers, therefore all integer values must first be converted to floating point. The following is an example of a routine using the interpreter arithmetic routines:

calculation C = (A+22) / (B*5)

Where values A and B are both positive unsigned 16 bit integer values these are both input from the keyboard at the start of the beginning of the routine and the result C is a five byte floating point value which is both stored in memory and displayed on the screen.
Variable storage locations in memory used by this routine are:

$C000 — lsb of value A
$C001 — msb of value A
$C002 — lsb of value B
$C003 — msb of value B
$C004 to $C008 — temporary floating point value storage 1
$C009 to $C00D — temporary floating point value storage 2
$C00E to $C012 — floating point result C storage

This article is extracted from the following books and readers are recommended to consult them for further information — *Advanced Commodore 64 Basic Revealed* and *Commodore 64 ROMs Revealed* both by Nick Hampshire and published by Collins.

# Scratchpad

**This month K Frost provides a couple of very handy routines for use on all machines.**

HOW OFTEN HAVE YOU wanted to put a scrolling message accross your screen? You know the type, they are used in most games programs to give instructions or a witty message. The first routine does just this. It is written in Basic but nevertheless is quite fast and would be very easy to include in your own programs as a subroutine.

All the routine requires is that the message to be scrolled is held in the string A$ and the positioning of the string is held in D$, i.e. D$ should hold a home and a number of cursor movements.

The 18 in the MID$ statement is the width of the message window. This can be any size but don't forget if you go over 40 the message will scroll over more than one line on the screen.

## All Things Bright

The second routine is one that will display a message and flash the letters of that message in different colours. This is very good for messages such as 'PRESS ANY KEY TO CONTINUE' or 'SPACE TO PLAY'.

Again the program is in Basic and you can easily add it as a subroutine to your own programs. The message that you wish to colour should be held in the string A$. $ holds all the colours through which you wish the letters to cycle. Do experiment with this as some very interesting effects can be achieved.

**PROGRAM: COLOUR / K.FROST**

```
100 PRINT"[CLEAR]":REM COLOURS
     K.FROST 1985
110 POKE 53280,0:POKE 53281,0
120 REM A$ IS THE STRING TO DISP
    LAY
130 A$="YOUR COMMODORE"
    :B$="[WHITE,RED,CYAN,MAGENTA,
    GREEN,BLUE,YELLOW,C2,C3,C4,C5,
    C6,C7,C8]"
140 REM * MAIN ROUTINE *
150 FOR A=0 TO 15:PRINT"[HOME,
    DOWN2,RIGHT5]";:REM POSITION
     THE STRING
160 FOR B=1 TO LEN(A$)
    :C=INT(LEN(A$)*RND(1))+1
    :PRINT MID$(B$,C,1)MID$(A$,B,
    1);
170 FOR X=0 TO 15:NEXT X,B,A
```

**PROGRAM: SCROLL/ K.FROST**

```
100 PRINT"[CLEAR]":REM SCROLLING
     K.FROST 1985
110 REM A$ IS THE STRING THAT
     YOU WANT TTO SCROLL
120 A$="THIS IS AN EXAMPLE OF
     SCROLLING FOR YOUR COMMODORE
    'S SCRATCHPAD"
130 REM D$ IS USED TO POSITION
     THE MESSAGE ON THE SCREEN
140 REM CHANGE THIS TO SUIT YOUR
     OWN NEEDS
150 D$="[HOME,DOWN5,RIGHT11]"
160 REM * MAIN ROUTINE *
170 B$=CHR$(29):C$=B$+B$+B$+B$
180 A$=C$+C$+C$+C$+A$+" "
    :FOR A=1 TO LEN(A$)
190 REM THE 18 IN THE NEXT LINE
     IS THE WIDTH OF THE MESSAGE
     WINDOW.
200 REM CHANGE THIS TO SUIT YOUR
     OWN NEEDS.
210 PRINT D$;MID$(A$,A,18);
    CHR$(145)
220 FOR X=0 TO 80:NEXT X,A
```

# ·B·B·B·BOUND TO BE A HIT···

"10 levels of fun make this torture excellent value for money. Nova rating. Definitely one of those 'just another go' games. Game of the month February" – Computer Gamer

"The most compulsive game I've ever played. If you don't buy it you'll never know what you've missed." says Gary Penn, Zzap! 64. Gold Medal Award. 97% overall.

**£9.95** cassette

**£12.95** disc

Free with Bounder Metabolis.
Is it a man, is it a bird?
(Commodore 64/128 version only)




# B·B·B·B·BOUNDER···

Gremlin Graphics Software Limited, Alpha House, 10 Carver Street, Sheffield S1 4FS. Tel: (0742) 753423

**Eric Doyle shows you how to foil the program pinchers.**

# STOP THIEF

AFTER SPENDING MANY A sleepless night and cloistered day creating and debugging your latest computer masterpiece, it's disconcerting to know that any Tom, Dick or Harriette can rip it off in seconds flat. In the past many methods have been suggested to prevent LISTing but few are satisfactorily secure.

The four most common methods prevent the list function from operating properly but only one of these works after the program has been RUN.

Firstly, there is the simple expedient of using a shifted 'L' in a REM statement on the first line of your program:

```
10 REM [S L]
```

Trying to list to a printer causes it to hang up after the REM and the normal command LIST merely produces the following:

```
10 REM
SYNTAX ERROR?
READY
```

To undo this protection it is merely a case of deleting line 10 and then the program can be freely listed by anyone. Not very secure.

The second method only protects one line of a listing and also uses the protection of the REM statement. The easiest way to see this in operation is to enter a line such as

```
10 PRINT"HELLO":REM"
```

Press the return key to enter the line and then move the cursor to the space after the quotes. Press the shifted delete key (INST) four times and then press it again four times unshifted (DEL). This should give four reversed letter T symbols. Next type GOTO 30 and press return.

If you now LIST the one line program you should see

```
10 PRINT"HELLO":GOTO 30
```

As there is no line 30 an error message would be expected but when this program is RUN it executes normally with no error. All that has happened is that the REM part of the line has been masked by the delete symbols and the GOTO is still seen by the operating system as being within a REM statement. When the system tries to LIST to the screen the deletes are executed and this effectively pulls back the GOTO over the REM. On a printer the trick is revealed because each delete is shown in its original form as a reversed T.

Instead of the intricate INST/DEL routine, just pressing RVS ON and the letter T gives the same effect but with less fiddling about.

Adding more deletes pulls the GOTO further back along the line and experimentation will show that the command can be pulled back over the line number and even on to the previous line.

This is useful because it can be used in conjunction with the first list protection method to disguise its presence.

```
17 PRINT"HELLO"
23 REM"[47 deletes]100
PRINT""[S L]
```

This apparently only lists a line 100 on the screen and gives a SYNTAX ERROR? message. The false line 100 would succeed in putting most people off the scent and using line numbers which are not divisible by 10 would make deletion of the coded lines difficult. On a printer the trick is revealed as the deletes are shown in their original form as reversed Ts.

Instead of the PRINT statement in line 17 you could use a PEEK statement which looks for the shifted L in line 23. To find the location of this character insert the following line:

```
17 IF PEEK(XXXX)<>204
THEN SYS 64738
```

Now enter the following in direct mode (no program line number)

```
FOR A=2049 to 5000:IF
PEEK(A)<> 204 THEN
NEXT
```

When the cursor reappears type PRINT A and re-enter line 17 with the number obtained (2140) in place of (XXXX). Repeat this line somewhere deep in the rest of your listing and hide it using the next method of protection. Remember that whatever line number is used for the two lines at the beginning of the program, the position of the shifted L will not move.

The third method takes advantage of the way a line of Basic is seen by the operating system. A line consists of two bytes which give the memory address of the start of the next line, two bytes giving the current line number and then the tokenised code for the Basic instructions followed by a null (zero) byte denoting the end of the line. When a line is listed the null byte is used to tell the system to start a new line, not the two byte pointer to the beginning of the next line. We can fool the system into jumping to the next line during a list by inserting a null byte where it least expects it causing the list to prematurely jump to the next line without listing the Basic code in the current line.

After writing your program, decide which line you want to hide and place any five letters at the beginning of the line.

```
10 SSSSSPRINT"HELLO"
```

Next insert a STOP at the end of the previous line or insert a new line which consists solely of a STOP command:

```
5 STOP
10 SSSSSPRINT"HELLO"
```

RUN the program until the stop is reached and the familiar break message is displayed. At this point the system has stored the memory location of the next line just in case CONT is used. Locations 61 ($3D) and 62 ($3E) contain these pointers so the start of the line is given by the formula PRINT PEEK (63)*256+PEEK(64). In the example the value would be 2055.

To avoid upsetting the line link and the line number add five to this value and poke the location given with zero (POKE2060,0). The dummy STOP command can then be deleted and a LIST will show only the line number of the hidden line whether listed to the screen or a printer.

For the final method of list protection we need to know a little about the way in which the 64's memory is organised. Locations 768 ($300) to 819 ($333) mainly consist of jump vectors for some of the main ROM routines like LOAD, SAVE, BREAK and, more importantly, LIST. A vector is a two byte number which gives the location of the start of the in-built machine code routine which performs the relevant task. For example, a memory map gives the LIST vectors as being in location 774 and 775 ($306–$307). Change either of the values found in these locations and the LIST function will be disabled causing all

PROGRAM: BASIC LOADER

```
5 PRINT"[CLEAR,DOWN]LOADING PLEASE WAIT";:FOR A=0 TO 86
   :D=0
10 FOR B=0 TO 7:READ C:D=D+C:POKE 49152+A*8+B,C:PRINT".";
   :NEXT
15 READ C:IF C<>D THEN PRINT:PRINT"[DOWN]
   ERROR IN LINE"A*10+100:END
20 NEXT:PRINT:PRINT"[DOWN]ALL OK":FOR A=0 TO 1000:NEXT
25 INPUT"[CLEAR,DOWN6]ARE YOU SAVING TO TAPE OR DISK (T/D
   )";SD$
30 IF SD$<>"T" AND SD$<>"D" THEN 25
35 IF SD$="T" THEN GOSUB 60:POKE 186,1
40 IF SD$="D" THEN GOSUB 70:POKE 186,8
45 PRINT"[DOWN2,SPC7,RVSON,SPC]PRESS <SPACE> WHEN READY
   [SPC,DOWN2]":PRINT"[SPC6]";
50 GET A$:IF A$<>CHR$(32) THEN 50
55 SYS 49802:END
60 PRINT"[DOWN]ENSURE TAPE IS CUED TO THE END OF THE[SPC3]
   'STOP THIEF!' BASIC PROGRAM."
65 RETURN
70 PRINT"[DOWN]PLACE YOUR 'STOP THIEF!' DISK IN THE[SPC4]
   DISK DRIVE."
75 RETURN
100 DATA 160,0,162,45,169,241,32,5,814
110 DATA 193,192,40,208,247,185,0,194,1259
120 DATA 170,185,216,193,32,5,193,192,1186
130 DATA 80,208,242,162,98,169,242,32,1233
140 DATA 5,193,192,120,208,247,32,14,1011
150 DATA 193,169,1,166,251,160,255,32,1227
160 DATA 186,255,169,16,162,138,160,194,1280
170 DATA 32,189,255,169,0,32,213,255,1145
180 DATA 32,14,193,169,193,133,254,160,1148
190 DATA 48,166,252,224,8,208,6,169,1081
200 DATA 0,133,253,240,4,169,80,133,1012
210 DATA 253,162,0,177,253,41,63,157,1106
220 DATA 160,4,232,200,192,128,208,243,1367
230 DATA 162,0,160,0,185,208,193,157,1065
240 DATA 24,5,232,200,192,25,208,244,1130
250 DATA 32,159,255,32,228,255,201,13,1175
260 DATA 208,246,166,252,224,8,240,5,1349
270 DATA 160,46,32,30,248,32,14,193,755
280 DATA 169,3,141,39,3,169,81,141,746
290 DATA 38,3,160,0,185,96,194,153,829
300 DATA 81,3,200,192,42,208,245,234,1205
310 DATA 234,234,234,169,100,160,0,153,1284
320 DATA 150,5,200,192,28,208,248,160,1191
330 DATA 0,185,239,193,153,190,5,200,1165
340 DATA 192,11,208,245,160,0,185,80,1081
350 DATA 194,41,63,73,128,153,201,5,858
360 DATA 200,192,16,208,241,169,160,153,1339
370 DATA 201,5,169,0,32,144,255,169,975
380 DATA 1,166,252,160,1,32,186,255,1053
390 DATA 169,58,162,80,160,194,32,189,1044
400 DATA 255,169,22,133,43,169,3,133,927
410 DATA 44,169,43,166,174,164,175,32,967
420 DATA 216,255,76,226,252,153,0,4,1182
430 DATA 138,153,112,7,200,96,169,0,875
440 DATA 133,254,169,4,133,255,162,3,1113
```

manner of things to occur if the command LIST is used. Inserting a line at the beginning of your program such as:

10 POKE768,226:POKE769,252

would result in a system reset if LIST was attempted. Changing these values to 131 and 164 respectively would apparently cause nothing to happen.

One word of caution, I'm all in favour of experimentation but be warned: never play around with vectors if you have something valuable in the memory, you'll lose it!

All these systems have a common fault: they are all easily circumvented once located. What is needed is a program which will run automatically and again we need to look at the vectors in the light of how the operating system leaves the LOAD command.

A call is made to the vector jump at location 770 ($302) which causes a warm reset of the system. If this vector is changed to cause the newly loaded program to run we will have achieved our aim. But how can this be done?

First we must check that the memory locations which define the start and end of the program are correctly primed and that the text pointer is set to the beginning of the program. Fortunately, this can be done by calling up a routine in the Basic ROM located at 42585 ($A659) and then a call to 42946 ($A7AE) will cause the execution of the program. This means that we have to place a short routine into the memory and point the warm start vectors to the start of the routine.

Finding a place to store the autorun routine can be fraught with problems because it must stay in memory to keep the program re-running.

The cassette buffer must be avoided in case the program needs to access the recorder for any reason. Just below this buffer are eight consecutive bytes of free memory which will house our six byte routine comfortably:

```
0334 JSR $A659
0337 JMP $A7AE
```

Two bytes of the warm start vector can now point to this routine and because the warm start is called up by the RUN/STOP-RESTORE routines the use of the RUN/STOP key will result in the program restarting from the beginning.

To ensure that the autorun will work, a wedge, called from the CHROUT vectors at 806($326), must be placed into the cassette buffer to alter the vector at 770($302) at the end of loading. When the system tries to print READY, it jumps into the wedge which changes the warm start vectors. All of the memory from 316 to the end of your Basic program is saved after the CHROUT vectors have been changed.

Obviously, you cannot change the vector without using a machine code SAVE routine. The one included here starts off with a Basic program which stores the details of the load and save devices and filenames. This jumps into a machine code routine which loads the program which you want to protect, devises a suitable loading screen and places the autorun details in memory. A save is then performed and a cold reset is performed allowing you to try your new autoloading program out.

The Basic loader program includes a save routine. To set up your Stop Thief! master enter and save the loader program on a spare tape or disk. Type in and save the Stop Thief! Master program and then load and run the Basic loader (so that it stores the program immediately after the Master program if you are using tape).

When using Stop Thief!, load the Master program and run it. This automatically loads the machine code program. Although you will receive a prompt, make sure that the tape/disk containing the program ready for conversion is in the recorder/drive.

After loading, suitable prompts will appear to enable you to successfully save your autorun program and disk users should not be too concerned about the fact that the saving screen proclaims that it is loading the program. Remember that this screen will be saved along with your program and therefore will become your loading screen.

The results of your labours will be unbreakable Basic programs. . .well almost!

```
450 DATA 160,120,169,32,145,254,200,208,1288
460 DATA 249,230,255,202,208,246,145,254,1789
470 DATA 200,192,112,208,249,96,0,0,1057
480 DATA 69,78,83,85,82,69,32,84,582
490 DATA 72,65,84,32,84,72,69,32,510
500 DATA 68,73,83,75,32,73,78,32,514
510 DATA 89,79,85,82,32,68,73,83,591
520 DATA 75,32,68,82,73,86,69,32,517
530 DATA 73,83,32,82,69,65,68,89,561
540 DATA 32,84,79,32,84,65,75,69,520
550 DATA 32,84,72,69,32,80,82,79,530
560 DATA 84,69,67,84,69,68,32,80,553
570 DATA 82,79,71,82,65,77,32,32,520
580 DATA 69,78,83,85,82,69,32,84,582
590 DATA 72,65,84,32,84,72,69,32,510
600 DATA 84,65,80,69,32,73,78,32,513
610 DATA 89,79,85,82,32,82,69,67,585
620 DATA 79,82,68,69,82,32,73,83,568
630 DATA 82,69,65,68,89,32,84,79,568
640 DATA 32,84,65,75,69,32,84,72,513
650 DATA 69,32,80,82,79,84,69,67,562
660 DATA 84,69,68,32,80,82,79,71,565
670 DATA 82,65,77,32,32,32,32,32,384
680 DATA 16,18,5,19,19,32,60,18,187
690 DATA 5,20,21,18,14,62,32,23,195
700 DATA 8,5,14,32,18,5,1,4,87
710 DATA 25,32,89,166,76,174,167,160,889
720 DATA 140,143,129,132,137,142,135,160,1118
730 DATA 160,160,160,0,0,0,0,0,480
740 DATA 160,170,160,170,160,170,160,170,1320
750 DATA 160,160,147,160,148,160,143,160,1238
760 DATA 144,160,160,148,160,136,160,137,1205
770 DATA 160,133,160,134,160,161,160,160,1228
780 DATA 170,160,170,160,170,160,170,160,1320
790 DATA 0,32,32,1,19,16,32,12,144
800 DATA 20,4,32,47,32,25,15,21,196
810 DATA 18,32,3,15,13,13,15,4,113
820 DATA 15,18,5,32,47,32,5,18,172
830 DATA 9,3,32,4,15,25,12,5,105
840 DATA 0,0,0,0,0,0,0,0,0
850 DATA 0,0,0,0,0,0,0,0,0
860 DATA 169,202,141,38,3,169,241,141,1104
870 DATA 39,3,160,6,185,116,3,153,665
880 DATA 51,3,136,208,247,169,0,133,947
890 DATA 198,169,52,141,2,3,169,3,737
900 DATA 141,3,3,96,32,89,166,76,606
910 DATA 174,167,169,1,166,186,160,1,1024
920 DATA 32,186,255,169,10,162,174,160,1148
930 DATA 194,32,189,255,169,0,133,253,1225
940 DATA 169,192,133,254,169,253,162,138,1470
950 DATA 160,194,32,216,255,96,84,72,1109
960 DATA 73,69,70,32,67,79,68,69,527
```

**PROGRAM: STOP THIEF!**

```
10 REM ***************************
20 REM *[SPC2]S T O P[SPC2]T H I E F ![SPC2]*
30 REM *[SPC24]*
40 REM *[SPC11]BY[SPC11]*
50 REM *[SPC7]ERIC DOYLE[SPC7]*
60 REM *[SPC24]*
70 REM * YOUR COMMODORE[SPC2]FEB 85 *
80 REM ***************************
90 GOTO 180
100 PRINT"[CLEAR,RVSON,GREEN,CE,C7,CE2,C3,CE2,C8,CE2,C6,
   CE2,YELLOW,CE22,C6,CE2,C8,CE2,C3,CE2,C7,CE2,GREEN,CE]";
110 PRINT"[C7,SPC]*[C3,SPC]*[C8,SPC]*[C6,SPC]*[YELLOW,
   SPC2]S T O P[SPC2]T H I E F ![SPC2,C6]*[SPC,C8]*[SPC,
   C3]*[SPC,C7]* ";
120 PRINT"[GREEN,CR,C7,CR2,C3,CR2,C8,CR2,C6,CR2,YELLOW,
   CR22,C6,CR2,C8,CR2,C3,CR2,C7,CR2,GREEN,CR]"
130 PRINT"[DOWN18]";
140 PRINT"[GREEN]-[C7]--[C3]--[C8]--[C6]--[YELLOW]
   ----------------------[C6]--[C8]--[C3]--[C7]--[GREEN]
   -";
150 PRINT"[C7]@[SPC,C3,SPC]A[C8]SP[C6,SPC]L[YELLOW]TD /
   YOUR COMMODORE / ER[C6]IC[C8,SPC]D[C3]OY[C7]LE";
160 PRINT"[GREEN,CI,C7,CI2,C3,CI2,C8,CI2,C6,CI2,YELLOW,
   CI22,C6,CI2,C8,CI2,C3,CI2,C7,CI2,YELLOW]";
170 POKE 2023,98:POKE 56295,5:RETURN
180 A=A+1:POKE 53280,9:POKE 53281,9
190 PN$="":PP$="":PAD$="[SPC16]":GOSUB 100:D=PEEK(186)
200 IF A<2 THEN PRINT"[HOME,DOWN7]LOADING CODE.[DOWN,SPC]
   PLEASE WAIT...":LOAD"THIEF CODE",D,1
210 INPUT"[HOME,DOWN5]LOAD ORIGINAL FROM TAPE OR DISK
   [SPC10,LEFT10]";TD$
220 TD$=LEFT$(TD$,1):IF TD$<>"T" AND TD$<>"D" THEN 210
230 POKE 251,1:LD$="TAPE":IF TD$="D" THEN POKE 251,8
   :LD$="DISK"
240 INPUT"[HOME,DOWN7]SAVE COPY TO TAPE OR DISK[SPC10,
   LEFT10]";TD$
250 TD$=LEFT$(TD$,1):IF TD$<>"T" AND TD$<>"D" THEN 240
260 POKE 252,1:SD$="TAPE":IF TD$="D" THEN POKE 252,8
   :SD$="DISK"
270 INPUT"[HOME,DOWN9]ORIGINAL NAME";PN$:IF PN$=""THEN 270
280 IF LEN(PN$)<17 THEN POKE 49203,LEN(PN$)
290 PN$=PN$+PAD$:PN$=LEFT$(PN$,16)
300 INPUT"[HOME,DOWN11]COPY NAME";PP$
   :IF PP$="*" THEN PP$=PN$
310 IF PP$="" THEN 300
320 IF LEN(PP$)<17 AND SD$<>"TAPE" THEN POKE 49385,
   LEN(PP$)
330 PP$=PP$+PAD$:PP$=LEFT$(PP$,16)
340 GOSUB 100
350 PRINT"[HOME,DOWN5,YELLOW]"PN$"[GREEN,SPC]
   WILL BE LOADED FROM"
360 PRINT"[DOWN,YELLOW]";LD$"[GREEN,SPC]AND SAVED IN ITS
   PROTECTED FORM ON"
370 PRINT"[DOWN,YELLOW]"SD$"[GREEN,SPC]UNDER THE NAME[SPC,
   YELLOW]"PP$
380 PRINT"[DOWN]IS THIS CORRECT?"
390 GET A$:IF A$<>"Y" AND A$<>"N" THEN 390
400 IF A$="N" THEN 180
410 FOR A=1 TO 16:POKE 49801+A,ASC(MID$(PN$,A,1))
   :POKE 49743+A,ASC(MID$(PP$,A,1))
420 NEXT
430 GOSUB 100:PRINT"[HOME,DOWN8]INSERT "LD$" CONTAINING
   "PN$
440 PRINT"PRESS <RETURN> WHEN READY"
450 GET A$:IF A$<>CHR$(13) THEN 450
460 SYS 49152
```

# COMPETITION

**Make magic with this month's prizes from Global Software.**



THERE'S MAGIC IN THE AIR AND mischief too in Magician's Ball from Global Software. If you want to improve your spelling then study the illustration on this page and see if you can be one of the lucky readers to enter an enchanted adventuring world. If you want to find out more about the game then turn to Sense of Adventure for details. There are 20 copies of the game to be won and the top five entrants will also receive a copy of Global's Caretaker program.

## How to Enter

STUDY THE ILLUSTRATION ON THIS page. Look very carefully as there are several enchanted balls hidden in the picture. When you think you have found all the hidden balls, circle them clearly.

Fill in the entry coupon carefully and seal it in an envelope, writing the number of balls you found on the back.

Send your entry to: Global Software Competition, Your Commodore, No 1 Golden Sq, London W1R 3AB. The closing date is Friday 28 February 1986.

You may enter as many times as you wish but each entry must be on an official coupon and sealed in a separate envelope.

**Important**: Please follow closely the guidelines on entering – incomplete coupons and entries with no numbers on the back cannot be considered.

## The Rules

ENTRIES WILL NOT BE ACCEPTED from employees of Argus Specialist Publications, Alabaster Passmore and Sons and Global Software. The restriction also applies to employees' families and agents of the company.

The How to Enter section forms part of the rules. The editor's decision is final and no correspondence will be entered into.

**Global Software Competition Entry Coupon**

**Name** ..............................................

**Address** ..............................................

..............................................

..............................................

.............................................. **post code**.........

**Number of Magician's balls found:** ..............................

Send to: Global Software Competition, Your Commodore, No 1 Golden Sq, London W1R 3AB. Write clearly and don't forget to write your answer on the back of your envelope.


THE COUNTDOWN HAS BEGUN
CRITICAL MASS
From DURELL
Commodore 64
Spectrum
R.R.P. £8.95
DURELL sales dept.,
Castle Lodge, Castle Green, Taunton TA1 4AB
Write now for details of Durell's
Great New Competition also
T shirts, Catalogue
and Posters

# MACH 4

**Steve Carrie adds a machine disassembler to the Mach 1 monitor.**

IN THIS ARTICLE, I WILL GIVE listings of the MACH 1 monitor extension and also some information on the whole MACH series of programs.

The extension adds a disassembler to the normal monitor commands. When the Basic program is RUN for the first time, it will patch the disassembler into the normal monitor code. Note that it replaces the 1 command (warm restart), so you will not be able to use the warm start facility (a similar effect can be created by issuing a G 8200 command).

When you have typed in and saved the extension listing, place a disk containing the original monitor program in the drive then RUN the program.

First, the extension code is placed in memory at address 7D00 hex. Each line has a checksum. If a data error occurs, the program will print the number of the line where it was detected. This checksum isn't 100% effective since one error may cancel out another so be careful!

Next, the original monitor code is loaded in at its usual address of 8200 hex. A series of POKEs patch the extension into the main code. These changes are as follows:

1 Change the I command to D and reset execution vector.
2 A section of code in the original monitor which sets the top of memory is altered to set at address 7D00 hex
3 Alter start-up message vector.

Finally, the whole program is saved to disk under the name of NEWMON. When you are sure that everything works OK, you can replace the original MONITOR file with this new version.

When the SAVE is complete, type SYS 64738 to reset the machine (don't switch the machine off!) then enter the monitor with SYS 33280 (unchanged). In addition to the usual start-up message, you should see another one similar to:

MACH1 EXTENSION
VERSION 3.5
(C) OCTOBER 1985 S.D.C.)Y.C.

Now, if you type D A560 <return>the disassembler should print out the contents of one of the ROM routines. Note that this command automatically sets hex I/O mode.

You should note that there is now about 1K less source code space for the editor which still leaves about 30K.

## The Monitor Jump Table

When I wrote the MACH series, I decided to put some of the more commonly used routines into the monitor. A jump table was provided at address 8200 hex to access these routines. Since the monitor is always present, a program only had to know where to call the required jump. A fair chunk of memory was saved using this method. In fact, looking back on it now, a lot more could have been saved.

The names of the routines and their call addresses are given below.

| Name | Address | Function |
|---|---|---|
| START | 8200 | WARM START |
| SAVE | 8203 | SAVE TO DISK |
| LOAD | 8206 | LOAD FROM DISK |
| FNAME | 8209 | PARSE FILENAME |
| OUTADD | 820C | OUTPUT A 2-BYTE NUMBER |
| OUTBYT | 820F | OUTPUT A 1-BYTE NUMBER |
| RET | 8212 | OUTPUT A CR/LF |
| EVAL | 8215 | EVALUATE EXPRESSION |
| CHKHEX | 8218 | CHECK ACC. FOR HEX CHAR. |
| CHKNUM | 821B | CHACK ACC. FOR NUMBER |
| CHKANU | 821E | CHECK ACC. FOR ALPHA-N |
| NSET | 8221 | SETS FILENAME PARAMETERS |
| COPEN | 8224 | OPEN DISK ERROR CHANNEL |

A JSR to the appropriate address should be used since each routine ends with RTS (except START).

Some are more useful than others. A more detailed description follows. Note that TXTPTR is the CHRGET text pointer at address 7A hex and TIB is the terminal input buffer at address 0200 hex.

**START** — This simply vectors to the code for restarting the monitor. It is a non-returning routine which destroys the 6502 system stack. This is normally used by an external command to return control to the monitor (see EXTERNAL COMMANDS). This is also the address 33280 used by the Basic start-up call.

**SAVE** — Save memory to disk. At entry, TXTPTR must point to an ASCII string in TIB which has the format:

filename < address1>,<address2>

This is the same format as the monitor S command. Any errors cause a jump to the error handler. On exit, TXTPTR points to the end of the string+1. This routine uses a logical file number of 1. Note that the Basic ROM is switched out to allow the area A000-BFFF hex to be saved too.

**LOAD** — Load from disk. On entry, TXTPTR points to an ASCII string in TIB with format:

<filename>

This is the same as the monitor L command. Any error causes a jump to the error handler routine. On exit, TXTPTR = end of string+1. Uses logical file 1.

**FNAME** — Parse filename. On entry, TXTPTR points to an ASCII string in TIB which represents the filename. At exit, registers are as follows:

X – length of filename string
Y – start offset of string in TIB
TXTPTR – end of string +1

This routine uses spaces as delimiters. An error is given if string exceeds 50 characters.

**OUTADD** — Output a 2-byte ASCII hex or decimal string. On entry, Y,X contain the 16-bit value to be printed. The output mode (hex or decimal) depends on the flag OUTMOD (address 02AE hex). If OUTMOD is zero, output mode is decimal otherwise it is hex.

**OUTBYT** — Similar to OUTADD except an 8-bit value

in the accumulator is used.

**RET** — Simply outputs a cr/lf combination to the current output device.

**EVAL** — Evaluation expression. On entry TXTPTR points to the start of the string. The flag OUTMOD operates in the usual way. Errors will be incurred if either the number is out of range (0<n>65535) or illegal characters are found.

Note that this routine also checks for the apostrophe (') which puts EVAL into ASCII mode regardless of OUTMOD. On exit, TXTPTR points to the end of the string +1.

**CHKHEX, CHKNUM, CHKANU** — These three routines check the accumulator for an ASCII hex, numeric or alpha-numeric character respectively. At exit, a carry-flag set indicates a valid character.

**NSET** — Set filename parameters. On entry TXTPTR points to the start of a filename in TIB. On exit, KERNAL routine SETNAM (FFBD hex) is called and TXTPTR = end of filename +1.

**COPE** — Opens disk device 8 error channel 15. No input parameters. There is no CCLOSE call. A file close may be accomplished using the following routine.

```
LDA #15 ; COPEN USES LFN 15
JSR $FFC3 ; KERNAL CLOSE
                     ROUTINE
```

Put simply, an external command is one whose code is not resident in the MACH1 area. Externals normally occupy the area from 9000 to 9FFF hex but may extend up to CFFF hex if the Basic ROM is switched out (if you use this method, remember to switch the ROM back in before calling START to return to the monitor).

When the external has finished executing, a JMP START allows the monitor to regain control. The actual call address of an external is 9000 hex.

The programs MACRO and ASSEMBLER are externals. Often, if the external you want to use has already been called (it must have been the last external called), a G 9000 command will provide a quicker method of execution. This is because an external remains resident after execution i.e. it is not deleted from memory. This is true until another external is called or the F command is used to clear out this section of memory.

## The Macro Processor Bug

And now... the bug! I discovered this while using the MACH system. It will only affect you if you use the micro processor on large source files containing macro calls. Basically, when the macro processor is constructing the output file, it uses the available memory from 0801 to 7CFF hex. Due to a programming oversight (a nice way of saying I made a mess) no check is made to see if this file is overflowing into the area above 7D00 hex. The processor will quite happily destroy itself!

If the application you are working on needs no macros then you can leave out the macro processing altogether and submit the raw code to the assembler.

Unfortunately, there is no easy way to tell when an overflow will occur. As a guideline, if your source code occupies more than about 25K and contains calls to some really big macros then you may run into trouble but I think about 90% of the time you will be OK. Sorry about that!

I have tried to make the MACH system reasonably versatile by including the external command facility. If anyone has any comments, questions or ideas on the system, I would be glad to hear about them. Write to Steve Carrie, c/o Your Commodore.

```
10 A=A+1:IF A=2 THEN 110
20 S=32000:PRINT"INSTALLING EXTENSION"
30 FOR L=200 TO 980 STEP 10
40 T=0
50 FOR B=0 TO 15
55 READ Z:POKE S,Z:S=S+1:T=T+Z
60 NEXT
70 READ TT:IF TT<>T THEN PRINT"DATA ERROR IN LINE";L:END
80 NEXT
90 PRINT"LOADING MONITOR."
100 LOAD"MONITOR",8,1
110 POKE 33340,68
120 POKE 33450,0:POKE 33451,125
130 POKE 35269,125
140 POKE 35284,76:POKE 35285,144:POKE 35286,129
145 PRINT"SAVING NEWMON"
150 POKE 43,0:POKE 44,125:POKE 45,128:POKE 46,142
160 SAVE "NEWMON",8,1
170 POKE 43,1:POKE 44,8:CLR
180 PRINT"FINISHED.":END
200 DATA 76,225,126,255,105,101,117,255,109,125,121,97,113,
   255,255,41, 2376
210 DATA 37,53,255,45,61,57,33,49,255,10,255,6,22,255,14,30,
   1437
220 DATA 255,255,255,255,255,255,36,255,255,44,255,255,255,
   255,255,255, 3650
230 DATA 201,197,213,255,205,221,217,193,209,255,255,224,
   228,255,255,236, 3619
240 DATA 255,255,255,255,255,255,192,196,255,255,204,255,
   255,255,255,255, 3907
250 DATA 255,255,198,214,255,206,222,255,255,255,255,255,73,
   69,85,255, 3362
260 DATA 77,93,89,65,81,255,255,255,230,246,255,238,254,255,
   255,255, 3158
270 DATA 255,255,255,255,255,255,76,255,255,255,255,108,255,
   255,255,255, 3754
280 DATA 255,32,255,255,255,255,255,255,169,165,181,255,173,
   189,185,161, 3295
290 DATA 177,255,255,162,166,255,182,174,255,190,255,255,
   255,255,160,164, 3415
300 DATA 180,255,172,188,255,255,255,255,74,255,70,86,255,
   78,94,255, 2982
310 DATA 255,255,255,255,9,5,21,255,13,29,25,1,17,255,42,
   255, 1947
320 DATA 38,54,255,46,62,255,255,255,255,106,255,102,118,
   255,110,126, 2547
330 DATA 255,255,255,255,255,233,229,245,255,237,253,249,
   225,241,255,255, 3952
340 DATA 255,133,149,255,141,157,153,129,145,255,255,255,
   134,255,150,142, 2963
350 DATA 255,255,255,255,255,255,255,255,132,148,255,140,255,
   255,255,255,255, 3735
360 DATA 66,67,67,66,67,83,66,69,81,66,77,73,66,78,69,66,
   1127
370 DATA 80,76,66,86,67,66,86,83,67,76,67,67,76,68,67,76,
   1174
380 DATA 73,67,76,86,68,69,88,68,69,89,73,78,88,73,78,89,
   1232
390 DATA 80,72,65,80,72,80,80,76,65,80,76,80,82,84,73,82,


   1227
400 DATA 84,83,83,69,67,83,69,68,83,69,73,84,65,88,84,65,
   1217
410 DATA 89,84,83,88,84,88,65,84,88,83,84,89,65,78,79,80,
   1311
420 DATA 66,82,75,65,68,67,65,78,68,65,83,76,66,73,84,67,
```

```
   1148
430 DATA 77,80,67,80,88,67,80,89,68,69,67,69,79,82,73,78,
   1213
440 DATA 67,74,77,80,74,83,82,76,68,65,76,68,88,76,68,89,
   1211
450 DATA 76,83,82,79,82,65,82,79,76,82,79,82,83,66,67,83,
   1246
460 DATA 84,65,83,84,88,83,84,89,144,176,240,48,208,16,80,
  112, 1684
470 DATA 24,216,88,184,202,136,232,200,72,8,104,40,64,96,56,
  248, 1970
480 DATA 120,170,168,186,138,154,152,234,0,0,11,22,33,44,55,
  66, 1553
490 DATA 77,88,99,110,121,132,143,154,165,176,187,198,209,
  220,231,242, 2552
500 DATA 255,169,255,141,174,2,32,121,0,240,3,32,21,130,32,
  201, 1808
510 DATA 128,32,18,130,32,228,255,240,251,201,13,208,241,
  108,2,3, 2090
520 DATA 165,90,201,1,144,7,201,5,176,10,76,28,127,169,1,
  133, 1534
530 DATA 91,76,70,127,201,8,144,18,201,10,240,14,169,2,133,
  91, 1595
540 DATA 160,1,177,20,141,1,2,76,70,127,169,3,133,91,160,1,
   1332
550 DATA 177,20,141,1,2,200,177,20,141,2,2,76,70,127,169,44,
   1369
560 DATA 44,169,32,76,210,255,32,65,127,32,65,127,173,0,2,
  32, 1441
570 DATA 15,130,166,91,224,1,240,19,173,1,2,32,15,130,166,
  91, 1496
580 DATA 224,2,240,13,173,2,2,32,15,130,96,32,65,127,32,65,
   1250
590 DATA 127,32,65,127,76,65,127,32,65,127,32,65,127,166,2,
  160, 1395
600 DATA 0,138,24,10,101,2,170,189,0,126,32,210,255,232,200,
  192, 1881
610 DATA 3,208,244,96,32,65,127,32,65,127,165,90,208,8,169,
  65, 1704
620 DATA 32,210,255,169,1,96,201,5,176,60,201,1,208,19,169,
  35, 1838
630 DATA 32,210,255,169,36,32,210,255,173,1,2,32,15,130,169,
  2, 1723
640 DATA 96,169,36,32,210,255,173,1,2,32,15,130,165,90,201,
  2, 1609
650 DATA 240,236,72,32,62,127,104,201,3,208,3,169,88,44,169,
  89, 1847
660 DATA 32,210,255,169,2,96,201,8,176,46,72,169,36,32,210,
  255, 1969
670 DATA 173,2,2,32,15,130,173,1,2,32,15,130,104,201,5,208,
   1225
680 DATA 3,169,3,96,72,32,62,127,104,201,6,208,3,169,88,44,
   1387
690 DATA 169,89,32,210,255,169,3,96,72,169,40,32,210,255,
  169,36, 2006
700 DATA 32,210,255,104,201,10,240,44,72,173,1,2,32,15,130,
  104, 1625
710 DATA 201,8,208,16,32,62,127,169,88,32,210,255,169,41,32,
  210, 1860
720 DATA 255,169,2,96,169,41,32,210,255,32,62,127,169,89,32,
  210, 1950
730 DATA 255,169,2,96,173,2,2,32,15,130,173,1,2,32,15,130,
   1229
740 DATA 169,41,32,210,255,169,3,96,177,20,141,1,2,169,2,
  133, 1620
750 DATA 91,32,70,127,32,119,127,32,65,127,32,65,127,165,20,
  24, 1255
760 DATA 105,2,133,253,165,21,105,0,133,254,173,1,2,48,14,
  24, 1433
770 DATA 101,253,133,253,165,254,105,0,133,254,76,178,128,
  169,0,56, 2258
780 DATA 237,1,2,133,3,165,253,56,229,3,133,253,165,254,233,
  0, 2120
790 DATA 133,254,165,254,32,15,130,165,253,32,15,130,96,24,
  101,20, 1819
800 DATA 133,20,165,21,105,0,133,21,96,169,0,133,2,133,90,
  133, 1354
810 DATA 91,32,65,127,165,21,32,15,130,165,20,32,15,130,32,
  65, 1137
820 DATA 127,160,0,177,20,141,0,2,201,255,240,45,162,0,221,
  168, 1919
830 DATA 126,240,8,232,224,33,144,246,76,75,129,134,2,224,8,
  176, 2077
840 DATA 9,200,32,104,128,169,2,76,189,128,169,1,133,91,32,
  70, 1533
850 DATA 127,32,119,127,169,1,76,189,128,169,1,133,91,32,70,
  127, 1591
860 DATA 32,65,127,32,65,127,169,66,32,210,255,169,89,32,
  210,255, 1935
870 DATA 169,84,32,210,255,32,65,127,32,65,127,169,36,32,
  210,255, 1900
880 DATA 173,0,2,32,15,130,169,1,76,189,128,162,0,221,3,125,
   1426
890 DATA 240,8,232,224,253,208,246,76,25,129,138,133,90,162,
  33,221, 2418
900 DATA 168,126,144,7,240,6,232,224,23,208,244,202,134,2,
  165,90, 2215
910 DATA 56,253,168,126,133,90,32,0,127,32,119,127,32,148,
  127,76, 1646
920 DATA 189,128,255,255,255,255,255,255,255,255,255,255,
  255,255,255,255, 3887
930 DATA 32,30,171,169,157,160,129,32,30,171,76,215,137,13,
  10,77, 1609
940 DATA 65,67,72,49,32,68,73,83,65,83,83,69,77,66,76,69,
   1097
950 DATA 82,32,69,88,84,69,78,83,73,79,78,13,10,86,69,82,
   1075
960 DATA 83,73,79,78,32,51,46,53,32,69,88,84,13,10,40,67,
   898
970 DATA 41,32,83,46,68,46,67,46,47,89,46,67,46,32,79,67,
   902
980 DATA 84,79,66,69,82,32,49,57,56,53,13,10,0,0,255,126,
   1031
```

■

# A GIFT 12 TIMES A YEAR

**FOR JUST . . .**

# £13

**NORMAL RATE . . . £16.60**

**A subscription to YOUR COMMODORE magazine is a gift twelve times over! Whether the subscription is for a friend, relative or a treat for yourself, YOUR COMMODORE provides the reader with the latest information and developments on the Commodore range of computers.**

**Delivered each month straight to the comfort of your own home, could a Commodore enthusiast wish for anything better?**

(This offer lasts until 28.2.86 and applies to readers in the United Kingdom only.)

*SUBSCRIPTION RATES*

Yearly overseas accelerated surface post: £21.50 or U.S.$28.00.
Yearly overseas air mail: £57.00

*Please return the completed form to:*

Your Commodore Special Offer
INFONET LTD
Times House
179 The Marlowes
Hemel Hempstead
Herts HP1 1BB

## Your COMMODORE

Please register the special £13.00 YOUR COMMODORE subscription below:

Name ..............................

Address ..............................

..............................

..............................

I would like to arrange a gift subscription to the person below: (Enter your own name and address above as donor.)

Name ..............................

Address ..............................

..............................

..............................

Please commence the subscription to YOUR COMMODORE with the ..................... issue.

I enclose payment of £.....................
(Cheques made payable to Argus Specialist Publications Ltd.)

Please charge my credit card account the amount of £..............

Card no. [ | | | | | | | | | | | | | ] Access VISA

Valid from .................... to ....................

Signature ..............................

Name ..............................

Address ..............................

..............................

..............................

# BREAK THE SPEED LIMIT

**This high speed tape operation for the C16 and Plus/4 will cut down that tedious waiting time. By Nick Hampshire.**

A FAST LOADER IS A ROUTINE WHICH replaces the existing LOAD and allows a program or data to be loaded from tape at about 10 times the speed of a normal LOAD so a tape can be as fast as a disk drive.

A fast loader is achieved by simply changing the format of the pulse sequence stored on the tape to allow a far greater density of information storage per inch of tape.

In order to create a fast load program two routines are needed. Firstly, a fast LOAD routine. This is a fairly short machine code routine loaded at the beginning of a LOAD operation and auto run to LOAD the rest of the program and/or data stored in fast loader format. The second program required is a routine to SAVE a program in fast loader format: the fast SAVE routine.

The first major problem to be overcome in designing a fast loader is how to store each bit on the tape. Each bit is stored on tape as a pulse which goes through a high-low transition (see Figure 1). The length of the total pulse decides whether the bit is a 1 or 0. A short pulse is a 0 and a long pulse is a 1. The bit is flagged in the interrupt register on the falling edge of the pulse.

The loader is a machine code program which runs with the interrupts disabled, sets a timer between the two lengths, and when the timer runs out the interrupt register is checked to see if the pulse came in or not. If the falling edge of the pulse generates an interrupt before the timer runs out then the pulse was a zero, if not, it was a one. The bits are then rotated into a byte storage until eight bits have been read, thereby loading a full byte.

Before any bytes can be read and stored, the loader must set itself to be in sync with the bits on the tape. This is done by writing a string of 0 bits with a single 1 bit at every byte interval. The routine then tries to align itself by recognising the value of the byte. An example of a header byte for aligning would be the value 64, hex $40 or in binary: 01000000. A series of these bytes is written as the header. Only when this byte has been read in and recognised can the actual program be read without risk of alignment errors.

The program is stored in different ways depending on how much program protection is desired. The simplest way of formatting the file is to first SAVE the two byte load address followed by the two byte end address and then the actual file. The final byte following the end of the file is a checksum calculated by the SAVE routine and it's also calculated during loading. If the two values are the same, the LOAD was successful. The routine for this form of fast loader is given in Program 1.

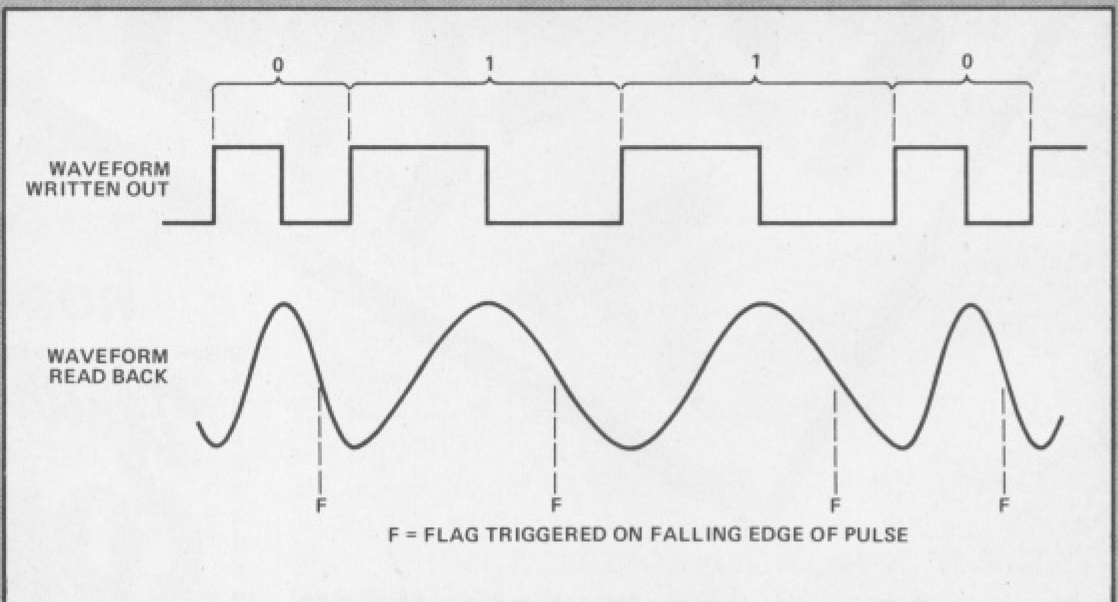



**Figure 1**

## Fast tape routines — making them work

Putting the theory into practice to create the fast LOADER routines is not difficult. The actual timing for the SAVE routine was not calculated from any theoretical formula but was obtained merely by trial and error. The only guidelines were that the short pulse should be slightly shorter than half the long pulse, since the waveform of the pulse is evened out by the cassette hardware. The timing value used by the loader is just shorter than the time required before the long pulse reaches its falling edge.

The high speed tape routine will SAVE a Basic program to tape in fast format and automatically put the fast LOAD routine into the filename where it is stored and, when loaded, will automatically start on the warm start vector. The routines are initialised by SYS(15616). A Basic program can be fast saved by using the SAVE command as normal but with a device number of 7, thus:

SAVE"PROGRAM",7

In addition the fast LOAD also makes use of the secondary address to auto run a program, thus:

SAVE"PROGRAM",7,1

This will cause the program to auto run when loaded back. With both routines, when a program has been saved using one of these fast loader SAVE routines it is unnecesary to LOAD anything before the program; it will LOAD directly from the LOAD command.

## Program Listing 1

```
033C           !FAST TAPE SAVE FOR THE COMMODORE 16.
033C           !***********************************
033C           !
033C           !THIS ROUTINE WILL SAVE A PROGRAM
033C           !TO TAPE SO THAT WHEN LOADED BACK
033C           !IT WILL LOAD NEARLY AS FAST AS
033C           !THE 1541 DISK DRIVE.
033C           !
033C           !AN OPTION FOR AUTO-RUN IS
033C           !INCLUDED.
033C           !
3D00           *=$3D00
3D00 A90B               LDA #<SAVVEC         !CHANGE SAVE VECTOR
3D02 8D3003             STA $0330            ! TO GO TO NEW
3D05 A93D               LDA #>SAVVEC         ! SAVE ROUTINE
3D07 8D3103             STA $0331
3D0A 60                 RTS
3D0B           !
3D0B           ! WEDGE INTO THE SAVE COMMAND
3D0B           !CHECKS FOR DEVICE #7 AND IF SO,
3D0B           !SAVE IN FAST FORMAT.
3D0B           !
3D0B 48        SAVVEC   PHA
3D0C A5AE               LDA $AE              !GET DEVICE #
3D0E C907               CMP #$07             !NUMBER 7?
3D10 F004               BEQ TSAVE            !YES
3D12 68                 PLA
3D13 4CA4F1             JMP $F1A4            !DO NORMAL SAVE
3D16           !
3D16           ! SAVE THE FILE. FIRST THE AUTO
3D16           !LOAD ROUTINE IS SAVED IN NORMAL
3D16           !TAPE FORMAT FOLLOWED BY THE FILE
3D16           !IN FAST FORMAT.
3D16           !
3D16 A5B2      TSAVE    LDA $B2              !MOVE SAVE ADDRESS
3D18 8D7B3F             STA STORE
3D1B A5B3               LDA $B3
3D1D 8D7C3F             STA STORE+1
3D20 A59D               LDA $9D
3D22 8D7D3F             STA STORE+2
3D25 A59E               LDA $9E
3D27 8D7E3F             STA STORE+3
3D2A A5AD               LDA $AD              !GET SEC. ADDR.
3D2C 8DBF3E             STA RUNFLG           !FLAG FOR AUTO-RUN
3D2F A00F               LDY #$0F
3D31 A920               LDA #$20
3D33 99C03E    LOOP1    STA FLNAME,Y         !BLANK FILENAME
3D36 88                 DEY
3D37 10FA               BPL LOOP1
3D39 A4AB               LDY $AB              !GET FILENAME LENGTH
3D3B C011               CPY #$11             !GREATER THAN 16?
3D3D 9002               BCC LOOP2            !NO
3D3F A010               LDY #$10             !ONLY 1ST 16 CHARS
3D41 88        LOOP2    DEY
3D42 3008               BMI TSAVE1
3D44 B1AF               LDA ($AF),Y          !GET FILENAME
3D46 99C03E             STA FLNAME,Y         !STORE IT
3D49 4C413D             JMP LOOP2            !DO NEXT CHAR
3D4C           !
3D4C A048      TSAVE1   LDY #FLNAME-LOADER
3D4E B9773E    TSAVE2   LDA LOADER-1,Y       !GET LOADER BYTE
3D51 99FF06             STA $06FF,Y          !STORE IT TO SAVE
3D54 88                 DEY
3D55 D0F7               BNE TSAVE2           !FOR ALL BYTES
3D57 2019E3             JSR $E319            !PRESS RECORD AND PLAY
3D5A A948               LDA #$48             !SET OUTPUT VECTOR TO
3D5C 8D2403             STA $0324            ! LOADER
3D5F A903               LDA #$03
```

## Program Listing 1 (cont.)

```
3D61 8D2503           STA $0325
3D64 A901             LDA #$01
3D66 AA               TAX
3D67 A8               TAY
3D68 20BAFF           JSR $FFBA          !SET FILE DETAILS
3D6B A9BC             LDA #$BC
3D6D A2C0             LDX #<FLNAME
3D6F A03E             LDY #>FLNAME
3D71 20BDFF           JSR $FFBD          !SET FILENAME DETAILA
3D74 A900             LDA #$00           !DISABLE 'SAVING'
3D76 859A             STA $9A
3D78 A903             LDA #$03           !SAVE $0324
3D7A 8523             STA $23
3D7C A924             LDA #$24
3D7E 8522             STA $22
3D80 A922             LDA #$22
3D82 A003             LDY #$03           ! TO $0326
3D84 A226             LDX #$26
3D86 20D8FF           JSR $FFD8          !SAVE IT
3D89 A94B             LDA #$4B           !RESET OUTPUT VECTOR
3D8B 8D2403           STA $0324
3D8E A9EC             LDA #$EC
3D90 8D2503           STA $0325
3D93 A900             LDA #$00
3D95 AA               TAX
3D96 A8               TAY
3D97 85B2             STA $B2
3D99 A907             LDA #$07           !FAST SAVE $0700
3D9B 85B3             STA $B3
3D9D 859E             STA $9E
3D9F A9B0             LDA #$B0           ! TO $07B0
3DA1 859D             STA $9D
3DA3 20C83D           JSR FSAVE          !SAVE IT
3DA6 A980             LDA #$80
3DA8 859A             STA $9A
3DAA A200             LDX #$00
3DAC A000             LDY #$00
3DAE AD7B3F           LDA STORE          !RESTORE SAVE ADDRESS
3DB1 85B2             STA $B2            ! OF MAIN FILE
3DB3 AD7C3F           LDA STORE+1
3DB6 85B3             STA $B3
3DB8 AD7D3F           LDA STORE+2
3DBB 859D             STA $9D
3DBD AD7E3F           LDA STORE+3
3DC0 859E             STA $9E
3DC2 20C83D           JSR FSAVE          !FAST SAVE MAIN FILE
3DC5 4C0387           JMP $8703          !EXIT TO 'READY.'
3DC8           !
3DC8           !THE FAST SAVE STARTS HERE
3DC8           !
3DC8 20123E FSAVE     JSR WRTHDR         !WRITE THE HEADER
3DCB A5B2             LDA $B2
3DCD 20403E           JSR WRTBYT         !LOAD ADDRESS LOW
3DD0 A5B3             LDA $B3
3DD2 20403E           JSR WRTBYT         !LOAD ADDRESS HIGH
3DD5 A59D             LDA $9D
3DD7 20403E           JSR WRTBYT         !END ADDRESS LOW
3DDA A59E             LDA $9E
3DDC 20403E           JSR WRTBYT         !END ADDRESS HIGH
3DDF 84B4             STY $B4
3DE1 A4B2             LDY $B2
3DE3 A900             LDA #$00
3DE5 85B2             STA $B2
3DE7 B1B2   TSAVLOOP  LDA ($B2),Y
3DE9 20403E           JSR WRTBYT         !ONE PROGRAM BYTE
3DEC C8               INY                !BUMP PROGRAM POINTER
3DED D002             BNE TSAVE3
3DEF E6B3             INC $B3
```

## Program Listing 1 (cont.)

```
3DF1 C49D    TSAVE3   CPY $9D          !REACHED END OF
3DF3 A5B3             LDA $B3          ! FILE?
3DF5 E59E             SBC $9E
3DF7 90EE             BCC TSAVLOOP     !NOT YET
3DF9 A5B4             LDA $B4
3DFB 20403E           JSR WRTBYT       !WRITE CHECKSUM
3DFE 20543E           JSR WRTBIT       !CLOSE OFF LAST BIT
3E01 A91B             LDA #$1B
3E03 8D06FF           STA $FF06        !RESTORE SCREEN
3E06 8D3EFF           STA $FF3E        !ROM BACK IN
3E09 A908             LDA #$08         !STOP TAPE
3E0B 8501             STA $01
3E0D 58               CLI              !START IRQ
3E0E 2084FF           JSR $FF84        !RESET I/O
3E11 60               RTS
3E12         !
3E12 78      WRTHDR   SEI              !DISABLE IRQ
3E13 8D3FFF           STA $FF3F        !ROM OUT
3E16 A900             LDA #$00         !START TAPE
3E18 8501             STA $01
3E1A A90B             LDA #$0B         !BLANK SCREEN
3E1C 8D06FF           STA $FF06
3E1F CA      HEADR1   DEX              !PAUSE FOR TAPE
3E20 D0FD             BNE HEADR1       ! DECK TO REACH SPEED
3E22 88               DEY
3E23 D0FA             BNE HEADR1
3E25 A9A0             LDA #$A0         !SET INITIAL TIMER
3E27 8D02FF           STA $FF02        ! VALUE INTO T2
3E2A A900             LDA #$00
3E2C 8D03FF           STA $FF03
3E2F A910             LDA #$10         !CLEAR TIMER
3E31 8D09FF           STA $FF09
3E34 A040             LDY #$40         !DO 64 TIMES
3E36 A910    HEADR2   LDA #$10
3E38 20403E           JSR WRTBYT       !WRITE VALUE 16
3E3B 88               DEY
3E3C D0F8             BNE HEADR2
3E3E A95A             LDA #$5A         !AND THEN CHAR 'Z'
3E40         !
3E40 85AB    WRTBYT   STA $AB          !STORE OTPUT BYTE
3E42 45B4             EOR $B4          !CALCULATE CHECKSUM
3E44 85B4             STA $B4
3E46 A908             LDA #$08         !LOOP 8 TIMES
3E48 85AC             STA $AC
3E4A 26AB    WBYTE1   ROL $AB          !BIT INTO CARRY
3E4C 20543E           JSR WRTBIT       !WRITE THE BIT
3E4F C6AC             DEC $AC
3E51 D0F7             BNE WBYTE1
3E53 60               RTS
3E54         !
3E54 A26C    WRTBIT   LDX #$6C         !ASSUME SHORT
3E56 9002             BCC WBIT1        !IS SHORT
3E58 A2FF             LDX #$FF         !ELSE SET LONG
3E5A 205D3E  WBIT1    JSR WBIT2        !SET OUTPUT HIGH
3E5D         ! AND THEN LOW
3E5D A910    WBIT2    LDA #$10         !WAIT FOR TIMER
3E5F 2C09FF  WBIT3    BIT $FF09
3E62 F0FB             BEQ WBIT3
3E64 48               PHA
3E65 A900             LDA #$00
3E67 8E02FF           STX $FF02        !RESET TIMER
3E6A 8D03FF           STA $FF03
3E6D 68               PLA
3E6E 8D09FF           STA $FF09        !CLEAR TIMER
3E71 A501             LDA $01          !TOGGLE WRITE BIT
3E73 4902             EOR #$02
3E75 8501             STA $01
3E77 60               RTS
```

## Program Listing 1 (cont.)

```
3040 3E78             !
3050 3E78             !THE LOADER STARTS HERE
3060 3E78             !
3070 3E78 A948   LOADER      LDA #$48           !SET RESET VECTOR
3080 3E7A 8DFCFF             STA $FFFC
3090 3E7D A903               LDA #$03
3100 3E7F 8DFDFF             STA $FFFD
3110 3E82 8D3FFF             STA $FF3F          !ROM OUT
3120 3E85 EE19FF             INC $FF19          !INCREASE BORDER COLOUR
3130 3E88 18                 CLC                !TELL LOADER TO 'RTS'
3140 3E89 204F03             JSR $034F          !LOAD A FILE
3150 3E8C 8D3EFF             STA $FF3E          !ROM IN
3160 3E8F A522               LDA $22
3170 3E91 8D19FF             STA $FF19          !RESET BORDER
3180 3E94 58                 CLI                !START IRQ
3190 3E95 208AFF             JSR $FF8A          !RESET OUTPUT VECTOR
3200 3E98 2084FF             JSR $FF84          !RESET I/O
3210 3E9B A59E               LDA $9E            !COMPARE CHECKSUMS
3220 3E9D C59D               CMP $9D
3230 3E9F D019               BNE LODERR         !DIFFERENT-'LOAD ERROR'
3240 3EA1 AD4707             LDA $0747          !RUN FLAG
3250 3EA4 F00E               BEQ EXIT           !CLEAR, RETURN TO READY
3260 3EA6 A9FF               LDA #$FF           !TURN OFF CURSOR
3270 3EA8 8D0CFF             STA $FF0C          ! (JUST IN CASE)
3280 3EAB 8D0DFF             STA $FF0D
3290 3EAE 20BE8B             JSR $8BBE          !PERFORM 'RUN'
3300 3EB1 4CDC8B             JMP $8BDC          !EXECUTE NEXT STATEMENT
3310 3EB4             !
3320 3EB4 209D8A EXIT        JSR $8A9D          !PERFORM 'CLR'
3330 3EB7 4C0387             JMP $8703          !'READY.'
3340 3EBA             !
3350 3EBA A21D   LODERR      LDX #$1D           !# FOR 'LOAD'
3360 3EBC 4C8386             JMP $8683          !OUTPUT ERROR
3370 3EBF             !
3380 3EBF 00     RUNFLG      BYT 0              !<>0 TO RUN PROGRAM
3390 3EC0             !
3400 3EC0             !*=$0337
3410 3EC0             !
3420 3EC0 202020 FLNAME      TXT "                 "
3430 3ED1             ! 17 SPACES
3440 3ED1             !*=$0348
3450 3ED1 AD19FF             LDA $FF19          !SAVE BORDER COLOUR
3460 3ED4 8522               STA $22
3470 3ED6 38                 SEC                !TELL LOADER TO 'JMP'
3480 3ED7 78                 SEI                !DISABLE IRQ
3490 3ED8             !*=$034F
3500 3ED8 08                 PHP                !SAVE STATUS
3510 3ED9 209303             JSR $0393          !READ THE HEADER
3520 3EDC 20B803             JSR $03B8          !READ LOAD LOW
3530 3EDF A8                 TAY                !INTO .Y
3540 3EE0 A900               LDA #$00           !ZERO LOW BYTE
3550 3EE2 85B2               STA $B2
3560 3EE4 20B803             JSR $03B8          !READ LOAD HIGH
3570 3EE7 85B3               STA $B3
3580 3EE9 20B803             JSR $03B8          !READ END LOW
3590 3EEC 852D               STA $2D
3600 3EEE 20B803             JSR $03B8          !READ END HIGH
3610 3EF1 852E               STA $2E
3620 3EF3 20B803 TLOAD1      JSR $03B8          !READ BYTE
3630 3EF6 91B2               STA ($B2),Y        !STORE IT
3640 3EF8 459E               EOR $9E            !CALCULATE CHECKSUM
3650 3EFA 859E               STA $9E
3660 3EFC C8                 INY                !INCREMENT POINTER
3670 3EFD D005               BNE TLOAD2         !NOT NEW PAGE
3680 3EFF E6B3               INC $B3            !INCREMENT HIGH BYTE
3690 3F01 EE19FF             INC $FF19          !INCREASE BORDER COLOUR
3700 3F04 C42D   TLOAD2      CPY $2D            !END OF FILE?
3710 3F06 A5B3               LDA $B3
```

## Program Listing 1 (cont.)

```
3F0A 90E7          BCC TLOAD1          !NOT YET
3F0C 20B803        JSR $03B8           !READ CHECKSUM
3F0F 859D          STA $9D             !STORE IT
3F11 A908          LDA #$08            !STOP TAPE
3F13 8501          STA $01
3F15 28            PLP                 !RESTORE STATUS
3F16 B001          BCS LOAD1           !CARRY SET, JMP $0700
3F18 60            RTS                 !ELSE 'RTS'
3F19 4C0007 LOAD1  JMP $0700
3F1C        !
3F1C        !*=$0393
3F1C        !
3F1C A900          LDA #$00            !START TAPE
3F1E 8501          STA $01
3F20 CA     RHEAD1 DEX                 !WAIT FOR TAPE DECK
3F21 D0FD          BNE RHEAD1          ! TO REACH FULL SPEED
3F23 88            DEY
3F24 D0FA          BNE RHEAD1
3F26 849E          STY $9E             !ZERO CHECKSUM
3F28 A200          LDX #$00
3F2A 20C603 RHEAD2 JSR $03C6           !READ BIT
3F2D 26AC          ROL $AC             !INTO BYTE
3F2F A5AC          LDA $AC
3F31 C910          CMP #$10            !VALUE 16?
3F33 D0F5          BNE RHEAD2          !NO
3F35 20B803 RHEAD3 JSR $03B8           !READ BYTE
3F38 C910          CMP #$10            !16?
3F3A F0F9          BEQ RHEAD3          !YES
3F3C C95A          CMP #$5A            !CHAR 'Z'?
3F3E D0EA          BNE RHEAD2          !NO, TRY AGAIN
3F40 60            RTS
3F41        !
3F41        !*=$03B8
3F41        !
3F41 A901          LDA #$01            !SET FOR 8 TIMES
3F43 85AC          STA $AC
3F45 20C603 GBYTE1 JSR $03C6           !READ BIT
3F48 26AC          ROL $AC             !INTO BYTE
3F4A 90F9          BCC GBYTE1
3F4C A5AC          LDA $AC             !LOAD BYTE
3F4E 60            RTS
3F4F        !
3F4F        !*=$03C6
3F4F        !
3F4F A910          LDA #$10
3F51 2401   GBIT1  BIT $01             !WAIT FOR INPUT
3F53 D0FC          BNE GBIT1           !LOW
3F55 2401   GBIT2  BIT $01             !WAIT FOR INPUT
3F57 F0FC          BEQ GBIT2           !HIGH
3F59 2D09FF        AND $FF09           !TEST TIMER
3F5C 48            PHA
3F5D A901          LDA #$01            !RESET TIMER
3F5F A278          LDX #$78
3F61 8E02FF        STX $FF02
3F64 8D03FF        STA $FF03
3F67 A910          LDA #$10            !CLEAR TIMER
3F69 8D09FF        STA $FF09
3F6C AD19FF        LDA $FF19           !INCREMENT BORDER
3F6F 18            CLC                 ! LUMINANCE
3F70 6910          ADC #$10
3F72 8D19FF        STA $FF19
3F75 68            PLA                 !GET TIMER STATUS
3F76 0A            ASL A               !SHIFT INTO CARRY
3F77 0A            ASL A               ! IF TIMER RAN OUT,
3F78 0A            ASL A               ! CARRY SET, BIT WAS
3F79 0A            ASL A               ! A '1' ELSE A '0'
3F7A 60            RTS
3F7B        !
3F7B 000000 STORE  BYT 0,0,0,0
```

## Program Listing 2 BASIC Loader

```
1000 REM ******************************
1010 REM * FAST SAVE ROUTINE FOR THE  *
1020 REM *COMMODORE 16. THIS ROUTINE  *
1030 REM *USES UP ONLY 758 BYTES OF   *
1040 REM *BASIC MEMORY.               *
1050 REM *                            *
1060 REM * COPYRIGHT 1985             *
1070 REM *    ZIFRA SOFTWARE LTD.     *
1080 REM *                            *
1090 REM ******************************
1100 POKE 51,0:POKE 53,0:POKE 55,0:POKE 52,61:POKE 54,61:POKE 56,61:CLR
1110 I=DEC("3D00"):T=0
1120 READ A:IF A=-1 THEN 1150
1130 POKE I,A:T=T+A
1140 I=I+1:GOTO 1120
1150 IF T<>74750 THEN PRINT"QQCHECKSUM ERROR :"T"SHOULD BE 74750":END
1160 IF I<>DEC("3F7B") THEN PRINT"QQNUMBER OF VALUES ERROR :"I"SHOULD BE 16251":
END
1170 PRINT"QQ DATA ENTERED CORRECTLY."
1180 PRINT"TO FAST SAVE A BASIC PROGRAM USE:"
1190 PRINT"QQ    SAVE"CHR$(34)"FILENAME"CHR$(34)",7"
1200 PRINT"Q OR QSAVE"CHR$(34)"FILENAME"CHR$(34)",7,1 TO AUTO-RUN."
1210 SYS DEC("3D00"):END
1220 DATA169,11,141,48,3,169,61
1230 DATA141,49,3,96,72,165,174
1240 DATA201,7,240,4,104,76,164
1250 DATA241,165,178,141,123,63,165
1260 DATA179,141,124,63,165,157,141
1270 DATA125,63,165,158,141,126,63
1280 DATA165,173,141,191,62,160,15
1290 DATA169,32,153,192,62,136,16
1300 DATA250,164,171,192,17,144,2
1310 DATA160,16,136,48,8,177,175
1320 DATA153,192,62,76,65,61,160
1330 DATA72,185,119,62,153,255,6
1340 DATA136,208,247,32,25,227,169
1350 DATA72,141,36,3,169,3,141
1360 DATA37,3,169,1,170,168,32
1370 DATA186,255,169,188,162,192,160
1380 DATA62,32,189,255,169,0,133
1390 DATA154,169,3,133,35,169,36
1400 DATA133,34,169,34,160,3,162
1410 DATA38,32,216,255,169,75,141
1420 DATA36,3,169,236,141,37,3
1430 DATA169,0,170,168,133,178,169
1440 DATA7,133,179,133,158,169,176
1450 DATA133,157,32,200,61,169,128
1460 DATA133,154,162,0,160,0,173
1470 DATA123,63,133,178,173,124,63
1480 DATA133,179,173,125,63,133,157
1490 DATA173,126,63,133,158,32,200
1500 DATA61,76,3,135,32,18,62
1510 DATA165,178,32,64,62,165,179
1520 DATA32,64,62,165,157,32,64
1530 DATA62,165,158,32,64,62,132
1540 DATA180,164,178,169,0,133,178
1550 DATA177,178,32,64,62,200,208
1560 DATA2,230,179,196,157,165,179
1570 DATA229,158,144,238,165,180,32
1580 DATA64,62,32,84,62,169,27
1590 DATA141,6,255,141,62,255,169
1600 DATA8,133,1,88,32,132,255
1610 DATA96,120,141,63,255,169,0
1620 DATA133,1,169,11,141,6,255
1630 DATA202,208,253,136,208,250,169
1640 DATA160,141,2,255,169,0,141
1650 DATA3,255,169,16,141,9,255
1660 DATA160,64,169,16,32,64,62
1670 DATA136,208,248,169,90,133,171
1680 DATA69,180,133,180,169,8,133
1690 DATA172,38,171,32,84,62,198
1700 DATA172,208,247,96,162,108,144
1710 DATA2,162,255,32,93,62,169
1720 DATA16,44,9,255,240,251,72
1730 DATA169,0,142,2,255,141,3
1740 DATA255,104,141,9,255,165,1
1750 DATA73,2,133,1,96,169,72
1760 DATA141,252,255,169,3,141,253
1770 DATA255,141,63,255,238,25,255
1780 DATA24,32,79,3,141,62,255
1790 DATA165,34,141,25,255,88,32
1800 DATA138,255,32,132,255,165,158
1810 DATA197,157,208,25,173,71,7
1820 DATA240,14,169,255,141,12,255
1830 DATA141,13,255,32,190,139,76
1840 DATA220,139,32,157,138,76,3
1850 DATA135,162,29,76,131,134,0
1860 DATA32,32,32,32,32,32,32
1870 DATA32,32,32,32,32,32,32
1880 DATA32,32,32,173,25,255,133
1890 DATA34,56,120,8,32,147,3
1900 DATA32,184,3,168,169,0,133
1910 DATA178,32,184,3,133,179,32
1920 DATA184,3,133,45,32,184,3
1930 DATA133,46,32,184,3,145,178
1940 DATA69,158,133,158,200,208,5
1950 DATA230,179,238,25,255,196,45
1960 DATA165,179,229,46,144,231,32
1970 DATA184,3,133,157,169,8,133
1980 DATA1,40,176,1,96,76,0
1990 DATA7,169,0,133,1,202,208
2000 DATA253,136,208,250,132,158,162
2010 DATA0,32,198,3,38,172,165
2020 DATA172,201,16,208,245,32,184
2030 DATA3,201,16,240,249,201,90
2040 DATA208,234,96,169,1,133,172
2050 DATA32,198,3,38,172,144,249
2060 DATA165,172,96,169,16,36,1
2070 DATA208,252,36,1,240,252,45
2080 DATA9,255,72,169,1,162,120
2090 DATA142,2,255,141,3,255,169
2100 DATA16,141,9,255,173,25,255
2110 DATA24,105,16,141,25,255,104
2120 DATA10,10,10,10,96,-1
```

# Teacher's Pet

**Margaret Webb goes back to school and learns about Basic.**

BEFORE LAUNCHING FORTH THIS month – a small confession. I've been writing this column for several months now and though my teaching qualifications are verifiable, the same cannot be said for my programming skills. I therefore decided that it was about time I rectified the situation by learning how to program in Basic. I soon discovered that this is not as easy as it would seem. The first question is to establish the best method of learning. Three approaches immediately spring to mind.

The first is to sign up at a local school or college for night classes. These are available for a number of languages with a bias towards BBC Basic. There are a number of snags however. Night classes are limited to the older age groups making them unavailable to youngsters. They also cost money.

If you want to use this route, you are constrained by a fixed timetable and term length. You must also learn what the teachers choose to teach. Some teachers have a strange idea as to what material a student needs. My sister is currently struggling through the mysteries of binary and hexadecimal as part of her course on Basic! If she was learning machine code I could understand it, but not for Basic. It's a good idea to find a course which offers some practical work as well as lectures so that you can get hands on experience with the help of the teacher.

A second approach is to find a home based course and work through it under your own steam. The alternatives are either a correspondence course, if you can find one, or a book/software package. The main drawback with this method is that if you get stuck, you don't have anyone to bail you out.

Alternatively, you could find a tutor. There are always knowledgeable enthusiasts around who would spend some time passing on information for a small sum. The problem is finding them.

Probably the best approach for most people is to combine the latter two.

This month I intend to look at a number of book/software systems currently available.

Most of you will have discovered the lack of help provided when you buy a Commodore machine. The C-16 does come with a Basic tutor but this is more of a sales gimmick. The actual manuals are next to useless. This is rather strange when you consider the tuition provided with other products. Many manufacturers of knitting machines and musical instruments provide lessons or other teaching material included in the price. Why can't computer manufacturers do it too?

One small word of warning. A large number of publishers have spotted the short comings of the Commodore manuals and have produced books teaching Basic and the inner workings of the computer. Many of these books are hardly any better than the Commodore manuals and you should purchase with care.

As a preliminary guide, I shall describe a number of packages covering different age-groups and prices.

Commodore offers a package for the Vic-20, C64, C-16 and Plus/4 called *An Introduction to Basic*. Whilst orginally sold as a separate package, it has subsequently been offered in the various Starter Packs. You may be able to locate it separately if you look hard enough or contact Commodore.

The package comprises of two cassettes and a book. The manual is set out as a series of experiments which are linked to programs on tape. The material ranges from setting the computer up to fairly advanced programming. I was pleased to see the frequent use of flow diagrams to demonstrate the operation of the programs. The approach is structured so that you must cover the early material before progressing through the manual. Overall it's quite a reasonable system which does its job well enough.

For the younger users, Collins offers a book entitled *DATA LOG*. This is a work book which has the appeal of a simple approach. Each page gently leads the user through the maze of programming, sprites and sound. The entire book is written in the form of a space ship's log with each section or exercise adding to the scenario. For example using screen printing and tabulation for the passenger list, IF...THEN for ticket details and using a sprite to design the captain's badge. The material is not covered in great depth but the book gives a good introduction to the use of Basic. At £2.95 it's also good value for money.

Glentop Publishers has a wide range of teaching books and book/software packages available for a number of Commodore machines. Originally starting with the Dr Watson series of books for Basic and machine code, Glentop moved on to the *Watson's Notes* series for the C64. This is a six book series which deals with most facets of the C64. The early volumes cover programming in Basic moving on to graphics in the later parts. The layout of the material is clear and tidy and is handled in a logical manner. If I do have a complaint it is that the content of the books is perhaps a little thin. At £2.95 per volume, I would have expected a little more depth. Notwithstanding this drawback, the series is well worth a close look.

Another good quality product from Glentop is Basic Adventure Part 1. This book/software package teaches the rudiments of Basic to the seven to 11 age group. The book takes the form of a science fiction story involving Dr Watson. Programming concepts are introduced in small routines which compliment the text. Overall a novel and effective way of camouflage teaching programming.

*Commodore 64 Basic* from Wiley is a meaty introduction to Basic. This is part of a Self Teaching guide series and uses self checking and exercises to aid you. The material covered is much as expected covering all facets of Basic, graphics and sound. Flowsheets are used extensively helping the pupil to develop a logical approach to programming. The book is sufficiently comprehensive that even once you have mastered programming in Basic, there are more advanced areas to look at. This is exemplified by the section on databases and data storage on disk and cassette. The author is clearly American and this does unfortunately become apparent in the text with the introduction of trans-Atlantic colloquialisms. Provided you can stand the literary style, it's book is pretty good value albeit a little pricey.

For C-16 owners, Shiva offers the *Gateway to Programming* series. This quartet of books tackles the subject in a similar manner to the Dr Watson series by using Sherlock Holmes stories to illustrate ideas. The text is humorous with lots of cartoon illustrations. In spite of the £4.95 price per volume, these are good value for money and worth a look.

Dorning Kindersley produces a Commodore 64 Starter Pack consisting of books one and two of their Screen shots books and a typing tutor cassette. The screen shots are full colour pictures of listings as they appear on the monitor and illustrations of the results you can expect from the program. The books step gently through learning Basic and the cassette should help you to get to grips with the keyboard. The boxed set costs £15.95 but if the price is off-putting each book in the set can be bought separately.

All of the books described here have the virtue of using a well thought out approach to teaching Basic. In addition most are cheap and almost within pocket money range.

# INTERRUPTS

**LK Gibson introduces C64 users to IRQ interrupts.**

## What is Multi-Tasking?

THIS TERM IS USED TO DEScribe a computer that can run more than one program at any given time, each program being transparent to the other. Output to the video monitor or TV screen is usually split so that windows are formed, each window being a miniature version of the normal screen for each program. This system of running programs concurrently can either be produced by hardware or software.

In the examples we are going to consider there are no windows and the ability to run two programs is derived purely from software.

Unfortunately this operation can only be performed in machine code, but don't despair, Basic programmers, the steps to have two programs up and running simultaneously are relatively simple and will be dealt with in as simple and concise a manner as possible.

## How it Works

The way in which we make a program run apparently transparent to anything that might be running is by making use of the system's IRQ INTERRUPTS, this interrupt is called 50 times every second (or once every jiffy).

When an IRQ occurs your C64 stops whatever it is doing, whether it's a Basic or a machine code program and goes off to carry out its own little machine code program Remember, this happens so fast that it is transparent to the system. The machine code, executed during an IRQ, is simply a housekeeping routine and does things like SCAN THE KEYBOARD, UPATE THE SYSTEM CLOCK etc.

When an IRQ occurs your C64 must know where to find this block of housekeeping code. It finds an answer from memory locations 788 and 789 (HEX $0314 & $315). The two numbers stored in these addresses form the INDIRECT starting address of the housekeeping code.

You may have noticed something here. Memory locations 788 and 789 are situated in RAM and that means that the information stored there can be changed at will. So before the processor tells the computer to go and do the housekeeping we redirect it to do whatever we want it to do first. As you now know, IRQ occurs 50 times every second so we now have a program that runs 50 times every second irrespective of whatever else your computer may be doing at the time. This forms the basis for running at least two programs concurrently. Now we shall go on to consider in more technical terms exactly how this process is achieved.

As stated earlier, memory addresses 788 and 789 contain the INDIRECT address for the start of the normal INTERRUPT CODE. Location 788 contains the low part of the INDIRECT ADDRESS and 789 forms the high part of the INDIRECT ADDRESS. This method of calculating INDIRECT ADDRESSES goes for any INDIRECT ADDRESS used by your computer, i.e. LOW BYTE/HIGH BYTE. Therefore, to find an actual address, we can use the formula;

```
AD=PEEK (x) + PEEK (x+1)*256
```

Where x is the first location (LOW BYTE) and x+1 is the second location (HIGH BYTE), i.e. 788 and 789.

Using this formula we can calculate the actual address of the standard INTERRUPT CODE with;

```
AD=PEEK(788)+PEEK(789)*256:
PRINTAD
```

This will give the start address as 59953 ($EA31). Write down this address as we need to jump to it at the end of our own custom routine, if this is not done the system will not scan the keyboard and the result will be a system crash. Therefore the last instruction in our code must be;

```
JMP $EA31
```

All of this will become quite clear later on so don't worry if you're slightly confused at the moment. The programs given later are presented in such a way that they can be used by the novice computer owner, but will also form the basis of some more advanced INTERRUPT DRIVEN CODE for the more experienced amongst you.

An example of this is to form windows using RASTER INTERRUPTS effectively splitting the screen using one half for one program and the second half for your interrupt code, unfortunately this is beyond the scope of this article.

## How to Set up Interrupts

An ASSEMBLY LANGUAGE program to set INTERRUPTS would go as follows;

| Address | Opcode | Operand | Comments |
|---|---|---|---|
| $C000 | SE1 | | disable interrupts while setting up |
| $C001 | LDA | #$20 | set low byte of start address for code |
| $C003 | STA | $0314 | store it at 788 decimal |
| $C006 | LDA | #$C0 | set high byte of address for code |
| $C008 | STA | $0315 | store it at 789 decimal |
| $C008 | CLI | | allow interrupts again |
| $C00C | RTS | | return to Basic |
| $C020 | | | examples given later |
| | JMP | $EA31 | jump to standard interrupt code |

$C020 is where your code begins and JMP $EA31 is the last instruction in your program.

This code is fairly standard for setting up any interrupt driven software, although some programmers may wish to carry out some initialisation for their program during this setting up procedure. The techniques for doing this will become apparent in later examples.

The routine as it stands will be situated in memory addresses 49152 (HEX $C000), this is a 4K block of RAM situated above the Basic INTERPRETER. This is a convenient place away from the ravages of Basic, but almost any RAM location may be used providing you don't clash with Basic or the SYSTEM VARIABLES. Some useful locations are given below, although this list is by no means complete.

**$C000(49152) to $CFFF(53347):** This is the spare 4K RAM BLOCK situated above the Basic INTERPRETER. As it is split from the normal Basic RAM (2048-40959) it cannot be affected by Basic, therefore this is an ideal place for our INTERRUPT DRIVEN CODE.

**Basic RAM $800(2048) to $9FFF(40959):** Code can be placed at the top of Basic RAM but unless it is protected it will be overwritten by Basic STRING VARIABLES. To protect this area we must lower the top of Basic and also lower the bottom of STRING STORAGE. The pointers to

these are stored at 55-56 for TOP OF Basic and 51-52 for BOTTOM OF STRING STORAGE. To protect 256 BYTES for our interrupt code we would use the program line;

10 POKE52,PEEK(52)–1:POKE 56,PEEK(56)–1

To increase this to 512 BYTES of protected area we would substitute -2 for -1 in the above program line and so on for each 256 BYTES required.

Line 10 as it stands will give us 256 protected BYTES from location 40704($9F01) to 40959($9FFF) for our code.

**Cassette Buffer 828-1023:** This is the cassette buffer and is totally safe for disk users but anything written there will be overwritten by cassette LOAD and SAVE operations, therefore cassette users must be very careful when placing code in this area.

From the above examples it can be seen that the easiest and most convenient addresses to place code are from 49152 ($C000) onwards, therefore all the examples given will use these addresses.

## Making it Work

Each example that follows will be preceded by a description of the program and will outline its purpose, this will then be followed by a Basic program containing the necessary code in data statements (this will allow users without a machine code monitor to enter and run the programs given). Next will be given an assembly listing (for those amongst you with machine code monitors) and finally each example will be annotated to show you how it works.

## Fuzzy Border

The following program is the shortest example that I could think of. Although it doesn't really serve any practical purpose it does give dramatic example of how INTERRUPT DRIVEN CODE works. When this program is complete you will see no difference to the 64 screen but the exterior border will be flashing dramatically, and will continue to do so even while you enter or load and run other programs.

Note that all these examples can be switched off by using the RUN/STOP and RESTORE keys together.

## Basic Program 1

This is a complete Basic program and will automatically boot the code when run. Type it in exactly as shown, save it for security purposes and then run it and behold!.

**Disassembled Listing 1 (with machine code monitor)**

If you try to enter Basic code once a machine code program has loaded the chances are you will get an 'OUT OF MEMORY' error. There is a Basic BOOT program at the end of this article that will overcome this problem.

Hopefully you have now entered and fully understood Program 1, if this is not the case then I would strongly recommend that you go back and read the preceding paragraphs, which should by

| | | |
|---|---|---|
| $C000 | SEI | disable interrupts |
| $C001 | LDA #$20 | load acc with low byte of indirect address |
| $C003 | STA $0314 | store it in low byte of IRQ RAM vector |
| $C006 | LDA # $C0 | load acc with with high byte of indirect address |
| $C008 | STA $0315 | store it in high byte of IRQ RAM vector |
| $C00B | CLI | enable interrupts again |
| $C00C | RTS | return to Basic |
| $C020 | LDX # $00 | set counter for screen colours |
| $C022 | STX $D020 | store it in border colour address |
| $C025 | DEX | decrement colour counter (ie change the colour) |
| $C026 | BNE $C022 | is it done 255 times |
| $C028 | JMP $EA31 | if yes then jump to standard IRQ code before returning |

```
PROGRAM:  BASIC PROGRAM 1

5 REM BASIC PROGRAM 1
10 FOR A=0 TO 12:READ MC
   :POKE 49152+A,MC:NEXT
20 FOR A=0 TO 10:READ MC
   :POKE 49184+A,MC:NEXT
30 SYS 49152
40 DATA 120,169,32,141,20,3,
   169,192,141,21,3,88,96
50 DATA 162,0,142,32,208,202,
   208,250,76,49,234,0
60 PRINT"OK CODE NOW ACTIVE!
   !!"
```

If you have machine code monitor then enter the above code as shown, again you must always SAVE code before executing it, this cannot be stressed enough. Switch your 64 off and then on again, type LOAD"YOUR PROGRAM NAME",1,1 and hit RETURN. Once the code has completed loading type SYS 49152 to activate the code.

now hold a lot more meaning.

That introductory program wasn't too difficult was it, we'll move on to some more practical routines now, so sit down and prepare yourself to enter a fascinating sphere of computer programming.

## Using a Vic 1540 Disk Unit on the C64

If you've seen that bargain secondhand Vic 1540 Disk Drive and had to pass it up because you thought you couldn't use it on your C64 then this routine is for you. The 1540 disk drive will operate exactly the same as the 1541 when connected to a 64 apart from one major problem. It won't load programs without you first blanking out the screen and then opening it up again when loading is complete. This is very laborious and there are in fact two ways of overcoming this problem.

1 Buy a new 1541 ROM chip from Commodore for £24 and try to fit it yourself, or pay a dealer to fit it at extra cost, or

2 Use this routine!

To use the program F1 blanks the screen and F3 opens it up again, therefore to load a program use the following procedure;

1 Type LOAD"YOUR PROG",8
2 PRESS F1 and HIT RETURN
3 Once the red 'disk in use' light has extinguished press F3

It's as simple as that!

By the way a more esoteric way of overcoming the problem is to blank and open the screen automatically using the RAM LOAD and SAVE vectors, but again that is beyond the scope of this article, sorry.

## Basic Program 2

Remember to save program 2 before you execute it!

Once you have saved this program for getting it back in the computer and up and running are exactly the same as that for PROGRAM 1.

If you're still with us and have at least partly understood the procedure involved for the programs above, then you should by now be starting to realise some of the weird and wonderful things that can be achieved with Interrupt Driven Code.

We now come to our final program concerning interrupts and as you would expect it is also the most complex we have dealt with to date.

## Defined Function Keys

This program gives us defined function keys, the keys are defined as follows:

F1=Change border colour. This will step through each of the 15 possible colours individually.
F3=Change screen colour. This will step through each of the 15 possible screen colours individually.
F5=Repeat keys toggle. This will toggle between all keys repeat and cursor keys only repeat.
F7=Processor pacer. This allows Basic programs to be stopped in mid run and allows for easy de-bugging and also a freeze frame facility. It also works with program listings to allow easier reading. Please note this should also work with most

m/c programs providing: They don't alter the Interrupt vectors, they don't occupy memory from $C000 to approx $C090, they don't disable Interrupts.

This is by no means the most complex task that can be handled by Interrupts but that's all for now, maybe more some other time editor permitting!

## Basic Program 3 (including initialisation)

Always remember to save routines before running them as a crash can be fatal. To activate the above program just LOAD the RUN then enter SYS49152.

### Disassembled Listings 2

| | | |
|---|---|---|
| $C000 | SEI | DISABLE INTERRUPTS |
| $C001 | LDA #$20 | LOAD ACC WITH LOW BYTE OF INDIRECT ADDRESS |
| $C003 | STA $0314 | STORE IT IN LOW BYTE OF IRQ RAM VECTOR |
| $C006 | LDA $C0 | LOAD ACC WITH HIGH BYTE OF INDIRECT ADDRESS |
| $C008 | STA $0315 | STORE IT IN HIGH BYTE OF IRQ IRQ RAM VECTOR |
| $C00B | CLI | ENABLE INTERRUPTS AGAIN |
| $C00C | RTS | RETURN TO BASIC |
| $C020 | LDA $C5 | GET THE LAST PRESSED |
| $C022 | CMP #$04 | IS IT THE F1 KEY |
| $C024 | BNE $C031 | IF NO THEN JUMP TO NEXT TEST |
| $C026 | LDA $D011 | LOAD ACC WITH VIDEO CHIP REGISTER DECIMAL 53265 |
| $C029 | AND #$EF | CLEAR BIT 4 ie set it to zero |
| $C02B | STA $D011 | PUT IT BACK ie blank the screen |
| $C02E | JMP $EA31 | GO DO STANDARD IRQ CODE BEFORE RETURNING |
| $C031 | CMP #$05 | IS IT THE F3 KEY |
| $C033 | BNE $C03D | NO THEN JUMP TO LAST INSTRUCTION IN ROUTINE |
| $C035 | LDA $D011 | GET VIDEO CHIP REGISTER DEC 53265 |
| $C038 | ORA #$10 | SET BIT 4 set bit 4 to a 1 |
| $C03A | STA $D011 | PUT IT BACK IN THE REGISTER ie open up the screen |
| $C03D | JMP $EA31 | JUMP TO STANDARD IRQ CODE BEFORE RETURNING |

PROGRAM: BASIC PROGRAM 2

```
5 REM BASIC PROGRAM 2
10 FOR A=0 TO 12:READ MC
   :POKE 49152+A,MC:NEXT
20 FOR A=0 TO 31:READ MC
   :POKE 49184+A,MC:NEXT
30 SYS 49152
40 DATA 120,169,32,141,20,3,
   169,192,141,21,3,88,96
50 DATA 165,197,201,4,208,11,
   173,17,208,41,239
60 DATA 141,17,208,76,49,234,
   201,5,208,8,173
70 DATA 17,208,9,16,141,17,
   208,76,49,234
```

### Assembly Listing (Requires assembler to enter)

```
10 *49152                START ADDRESS
20 IRQVEC=788            HI BYTE INTERRUPT VECTOR
30 IRQOUT=$EA31          NORMAL IRQ VECTOR
32 LDA  #0
33 STA 650               SET NO KEY REPEAT
35 SEI                   DISABLE INTERRUPTS
40 LDA  #<JUMP           SET NEW IRQ VECTORS
50 STA IRQVEC
60 LDA  #>JUMP
70 STA IRQVEC+1
80 CLI                   ENABLE INTERRUPTS AGAIN
90 RTS                   RETURN TO BASIC
100 JUMP LDA 197         READ LAST KEY PRESSED
110 CMP  #64             IS IT NO KEY
120 BEQ QUIT             NO KEY THEN RETURN
122 CMP  #4              IS IT F1
124 BNE TEST2            NO THEN TEST FOR F2
126 LDY 53280            YES THEN GET CURRENT
                         BORDER COLOUR
127 INY                  ADD 1 TO IT
128 CPY # 16
130 BNE CONT1
132 LDY  #0              RESET IF 15
134 CONT1 STY 53280      STORE NEW VALUE IN
                         BORDER COLOUR ADDRESS
135 JSR DEBO             SET UP KEY DEBOUNCE GOTO
136 JSR TILOOP           DO KEY DEBOUNCE
138 JMP QUIT             JUMP TO END
140 TEST2 CMP  #5        IS KEY F2
142 BNE TEST3            NO THEN TEST SEE IF IT IS F3
                         KEY
144 LDY 53281            YES THEN GET CURRENT
                         SCREEN COLOUR
146 INY                  ADD 1 TO IT
148 CPY  #16
150 BNE CONT2
152 LDY  #0              RESET IF 15
154 CONT2 STY 53281      STORE IT BACK IN SCREEN
                         COLOUR ADDRESS
155 JSR DEBO             SEE ABOVE
156 JSR TILOOP           SEE ABOVE
158 JMP QUIT             JUMP TO END
160 TEST3 CMP  #6        IS IT F3
161 BNE TEST4            IF NOT THEN SEE IF KEY IS F4
162 LDA 650              GET CURRENT KEY REPEAT
                         STATUS
163 EOR # 255            DO BINARY INVERT (IE 0 TO
                         255 OR 255 TO 0)
164 STA 650              STORE IT BACK IN KEY REPEAT
                         ADDRESS
166 JSR DEBO             SEE ABOVE
168 JSR TILOOP           SEE ABOVE
170 JMP QUIT             JUMP TO END
180 TEST4 CMP  #3        IS KEY PRESSED F4
182 BNE QUIT             NO THEN ALL FKEYS TRIED
                         JUMP TO END
184 LDY  #255            SET UP MAX time FOR TIME
                         LOOP
```

```
185 STY 253
186 JSR TILOOP              GOTO TIME WASTE
                            SUBROUTINE
300 QUIT JUMP IRQOUTJUMP TO NORMAL IRQ
                            INTERRUPT SUBROUTINE
400 TILOOP LDY 253          TIME WASTING LOOP
410 LOOP1 LDX #255
420 LOOP2 DEX
421 NOP                     TIME WASTING NOP
                            OPCODES
422 NOP                     THESE CODES DO NOTHING
423 NOP                     BUT WASTE TIME. BECAUSE
424 NOP                     WE ARE WORKING IN VERY
425 NOP                     FAST MACHINE CODE WE
426 NOP                     NEED THEM TO MAKE UP
427 NOP                     A REALISTIC TIME DELAY.
428 NOP
429 NOP
430 BNE LOOP2
440 DEY
450 BNE LOOP1
460 RTS                     RETURN FROM SUBROUTINE.
500 DEBO LDY #50            THIS SUBROUTINE SETS UP A
                            SHORTER TIME DELAY
510 STY 253                 TO ENSURE THAT THE KEYS
                            DO NOT BOUNCE
540 RTS                     RETURN FROM SUBROUTINE
```

PROGRAM: BASIC PROGRAM 3

```
10 I=49152
20 READ A:IF A= 256 THEN END
30 POKE I,A:I=I+1:GOTO 20
49152 DATA 169,0,141,138,2,120,169,18
49160 DATA 141,20,3,169,192,141,21,3
49168 DATA 88,96,165,197,201,64,240,84
49176 DATA 201,4,208,22,172,32,208,200
49184 DATA 192,16,208,2,160,0,140,32
49192 DATA 208,32,131,192,32,111,192,76
49200 DATA 108,192,201,5,208,22,172,33
49208 DATA 208,200,192,16,208,2,160,0
49216 DATA 140,33,208,32,131,192,32,111
49224 DATA 192,76,108,192,201,6,208,17
49232 DATA 173,138,2,73,255,141,138,2
49240 DATA 32,131,192,32,111,192,76,108
49248 DATA 192,201,3,208,7,160,255,132
49256 DATA 253,32,112,192,76,49,234
49264 DATA 164,253,162,255,202,234,234,234
49272 DATA 234,234,234,234,234,234,208,244
49280 DATA 136,208,239,96,160,50,132,253
49288 DATA 96,256
```

Please remember all of the above programs, especially the final one were not written with either speed or memory usage as their main criteria, but above all they were to be simple and concise making legibility easier. There are always various ways to solve most programming problems and the ones given were not necessarily the best solutions.

All of the assembly listings where produced on Supersoft's Mikro assembler cartridge on the C64. Some assemblers may use slightly different assembler command codes. Conversion to these assemblers should be little or no problem due to the fact that the assembly listings are fully annotated. Of course should you have the Mikro cartridge you should have no problems at all.

## Assembler Listing Loader Programs

As was mentioned earlier, if the assembled versions of programs in this article are loaded using

LOAD "PROGRAM NAME",8,1

Then any attempt to load a Basic program will fail and give the Error OUT OF MEMORY. The best way to overcome this is by way of a small program called a BOOT PROGRAM an example of which is now given below;

## Tape Boot Program

10 IF A THEN 30
20 A=A+1:LOAD"INTERRUPT program",1,1
30 SYS 49152:NEW (OR SEE NOTE BELOW)

Disk Boot Program

10 IF A THEN 130
20 A=A+1:LOAD"INTERRUPT PROGRAM",8,1
30 SYS 49152:NEW (OR SEE NOTE BELOW)

PROGRAM: GEN BOOT

```
1 REM **********************************
3 REM *[SPC3]L.K.GIBSON 1983[SPC11]*
4 REM **********************************
10 POKE 53280,1:POKE 53281,1:POKE 646,6
15 IF Z2=1 THEN SYS H1+L1
20 PRINT "[CLEAR] ENTER FILENAME"
30 INPUT Z1$
31 PRINT:PRINT"SEARCHING FOR LOAD ADDRESS OK!"
32 PRINT:PRINT"PROGRAM WILL AUTOBOOT ON LOAD COMPLETE!"
34 OPEN 1,8,3,Z1$
36 GET#1,L1$
38 GET#1,H1$
40 Y1$=H1$:GOSUB 200:H1=X1*256
42 Y1$=L1$:GOSUB 200:L1=X1
44 PRINT"[DOWN2]";H1+L1:CLOSE 1
46 Z2=Z2+1:LOAD Z1$,8,1
50 SYS H1+L1
200 IF Y1$=""THEN X1=0:RETURN
210 X1=ASC(Y1$)
220 RETURN
```

## Note

Instead of NEW which will wipe out the Basic Loader and leave the Interrupt Code running, a Basic program (your Basic Program) could continue from here of which the Interrupt Program forms an integral part.

Any of the programs that Load with 8,1 at the end (this is called a Relocated Load) will work using the above Boot programs. Remember the value for the SYS command may have to be altered to accommodate different code entry points.

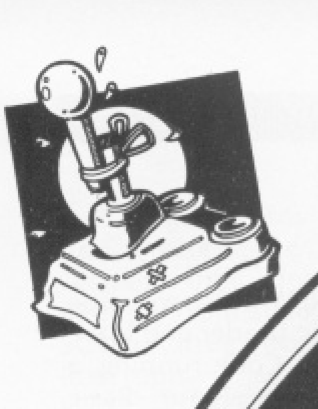
## A Little Utility

The program listed below is entirely in Basic and provides a useful facility. Should you ever come across a machine code program, whether it's one you have written yourself (and have forgotten the SYS entry point) or one written by somebody else, then this is for you. It can be LOADed using the 8,1 or the 1,1 suffix but how do you know where it resides in memory and what value do you use with the SYS command to activate it? Worry not, this program automatically Loads and Activates almost any machine code Program. The only programs will not activate are ones where the code entry point is not at the actual beginning of the machine code Program. Although it will still tell you where the Program resides so you can use a Monitor to find that out for yourself. Try it out on the programs contained in this article, they will all work. The utility is listed below and is called General BOOTstrap.

## The Program

We have now reached the end of this article, hopefully with a better understanding of how our computer works particularly IRQ Interrupts.

If you have found this article interesting or have any problems and you own a modem then please drop me a line on:
COMPUNET:LKG82.
SYSTEL:533848507.



# IN ARCADIA






**Flippo flips over Ancipital and gets hammered at chess.**

YOU KNOW WHAT? IT'S BEEN A HELL OF a month! I topped my high score on Anticipital (shuddup! It's a good game, actually!), finally found a good chess game that doesn't bore the pants off me, and not only that I've got a stack of hints'n'tips that'll knock yer eyes off!

## 'Cip it and See

OK, here goes. Ancipital is not really a new game, I admit that. But it's certainly one of Jeffy Minter's best. I get a bit tired of most of his other stuff; the early out'n'out shooty ups are fine for a few moments, but quickly pall in the end. The experimental Mama Llama makes my wrist hurt, and I find it more than a little bit frustrating, having a Killdroid careering all over the screen totally out of conscious control. (P'raps you should use the Force? — Ed.) Very funny!

Where was I? Oh, yes. Ancipital, or just plain 'Cippy to afficionados, is a true original. Four way gravity, a really off the wall objective, and a good slant on Jeff's alternative Universe. I like that. I think games which have a background story are more fun to play. You know, a little something to read before you power up the game, to get you into the feel of the story. That's where Elite scored very highly in my estimations, and that's in the little sci-fi book you get with the package. Come on, software houses! You're here to entertain us, so how about it.

What's that? What's my hi-score on 'Cippy? Nah, I'm too modest. Nah! Geddaway! Aw, alright, if you must know, it was 368,805. Pretty good, huh?

## Chess Mate

I must tell you about this... Look, I'm not really known for being a chess player. Well, actually, I'm rubbish, but I do keep playing it, despite humiliating defeats to man and machine alike. I dunno, it's akin to the fatal fascination some folk have with car crashes. The best chess game I've stumbled over recently, in my gluttony for defeat is Colosuss Chess 2.0. It's so easy to play; simply moving your pieces using the cursor, rather than the old long winded KP3-Kn4 kind of input you have to suffer in others I could mention. It's a delightful game, bringing back some of the peace and quiet of the real game. It beats the pants off me every time!

## Hint Me Daddio! (Eight to the Bar)

Here are a few tips for your notepad. Stop me if you've heard any of them before.

Jet Set Willy: This game drives me bonkers! I thought I'd seen the last of this on my monitor, but alas no; the game is back with new rooms, new challenges, and not a POKE in sight! Blast! I still can't leave it alone though. Shin up the rope in the Cold Store and you get into the Sewer System (poo!). Go all the way up the back stairs to Nomen Luni (Top Of Plane), skip across to On The Roof, Up On The Battlements and finally We Must Perform A Quirkafleeg. (OK, Matthew Smith, I know you read Furry Freak Brothers Comix!) Climb up the rope and you'll find yourself in the Watchtower. Get to the top, jump up, and you'll find yourself in the Rocket Room. Grab the gems at the top of the Rocket, and off you go, you're in the Space Station. Once on the Station, find your way to the Transporter, and you'll find yet another new system of rooms. There's also a guest appearance of a room from Manic Miner...good grief!

Rockfords Riot – Boulderdash II: On the first screen you have to blast a hole through a wall by dropping a rock on a firefly. Then you must clear the earth under a wall, and then drop rocks onto it. The wall is Magic, and it will create a gem for every rock that passes through it! General tips: The amoebas will create gems if they are contained by rocks. Try snatching gems from the side of piles of rocks and examine the way the rocks fall. There's an interesting clue on how to get seemingly enclosed gems out by utilising creative snatching. (Sounds painful, but never mind!)

More from Rockford next time.

## Raves from the Fave File

My current favourite games are Ancipital (Llamasoft), for reasons previously specified: Bounty Bob Strikes Back (US Gold) still my bestest platform game, beating the pants off Willy, any day; Summer Games II (Epyx/CBS) if only for the fencing and the Kayaking; Rescue on Fractalus (Lucasfilm/Activision) one of the most state of the art arcade games in existence, and if Jeff endorses it, then so do I; Ball Blazer (Lucasfilm) probably the most dizzying 3D game out, fast and furious; Rupert and the Toymakers Party (Quicksilva) Ahhhhh! Init sweet? And finally, Rock'n'Bolt (Activison), to my mind, the only new game from these boys worth a light.

## That's Yer Lot!

OK, that's all in the Saga of Yellow Prindle Walks a Long Way Off and Conquers His Fear of Strange Fruit...we have time for this month. Next time we'll... well actually you'd better wait and see, 'cos I always spill the beans, and what thanks do I get? None. So off you go and write me a letter about your high scores. Go on! And don't turn to the next page until you've done it!
Hurumph!

# £10,000 HOLIDAY BONANZA





Maintenance, Service, Support, Training all from your No 1 Commodore centre.

**WELCOME TO THE WORLD OF COMMODORE**

**COMMODORE 64**
computer compendium

Commodore 64 computer
cassette unit
music maker keyboard and software
designers pencil software
the secret diary of Adrian Mole software
+FREE the growing pains of Adrian Mole

£199.99

**PRINTERS**

| | |
|---|---|
| **DAISY WHEELS** | |
| DPS1101 | 199.99 |
| JVK1 6100 | 399.95 |
| DAISY STEP 2000 | 287.50 |
| **DOT MATRIX** | |
| LX 80 INC T/FEED | 293.00 |
| MP165 | 299.00 |
| FX80 | 399.95 |

**Free Seiko RC-1000**

If you buy either of the following

MPS803 dot matrix printer
1541 disk drive
1701 colour monitor

**£199.99 EACH.**
+ FREE Seiko RC-1000 wrist terminal/watch worth £99.99

**Special Package Deal**

C64 computer
1541 disk drive
Super Zaxxon game
Exodus – Ultima III game

£299.99

**C128 and 1570 Package**

C128 computer
1570 disk drive
5 × diskettes

£445.00

## Yes Its true !!!

**For a limited period we are giving away a £5.00 Travel Chek for spending money on your next holiday with EVERY £50 SPENT.**
**Thats not all folks !!!**
**your vouchers could also win you a FREE stay HOLIDAY for two or four people in our lucky numbers draw.**

**Peripheral Pack**

1541 disk drive
AND dot matrix printer

£199.99

**1570**

single sided disk drive
170K (formulated) storage capacity

£199.99

**64 & 128**

| Software | |
|---|---|
| Superscript | 85.00 |
| Superbase | 85.00 |
| Flight Sim II | 48.00 |
| & MORE | |

**MONITORS**

| | |
|---|---|
| 1701 COL | 199.99 |
| 1901 GREEN | 99.99 |
| 1902 COL | 299.99 |
| CUB 1431 AP | 258.95 |
| PHILLIPS GREEN | 79.99 |
| PHILLIPS CABLE | 10.00 |

**C128**

The 3 in 1 computer
100% compatible with C64 software
128K RAM, 72K ROM, 6510 & Z80A CPU's

£269.00

**Music to COMMODORE**
user ears

The exciting range of software for the 64 & 128

| | |
|---|---|
| Music Maker 128 | 29.99 |
| Play-a-long Albums (pop, Beatles or Classic) | 9.99 |
| Commodore sound studio | £14.95 |
| Sound Sampler | 69.99 |
| Sound Expander | 99.99 |

## CHROMASONIC
*Computer Centres*

48 Junction Road, Archway, London N19 5RD
01-263-9493/5
238 Muswell Hill Broadway, London N10 3SH
01-883-3705

**Epson Printer Pack**

LX80 NLQ printer
tractor feed unit
comprint interface & cable
(allows full Commodore graphics)
box of 2000 sheets 11 × 9.5 paper

£359.99



* CONDITIONS OF REDEMPTION OF TRAVELLERS CHEKS PRINTED ON EACH VOUCHER

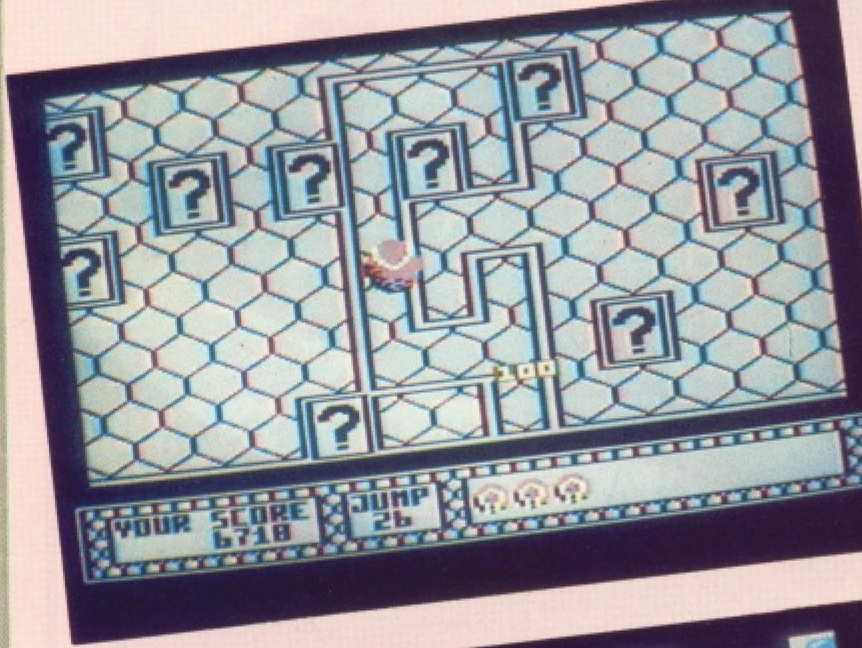
"INSTANT CREDIT" "INSTANT CREDIT"

**Terms**

All products are guaranteed for one year unless otherwise stated. Payment may be made by Access, Barclaycard, Bankers draft, Building Society cheque, cash or postal order. Sorry, cheques need five days for clearance. We reserve the right to change prices without prior notice. **All prices are inclusive of VAT.** Please check before ordering for carriage charges.

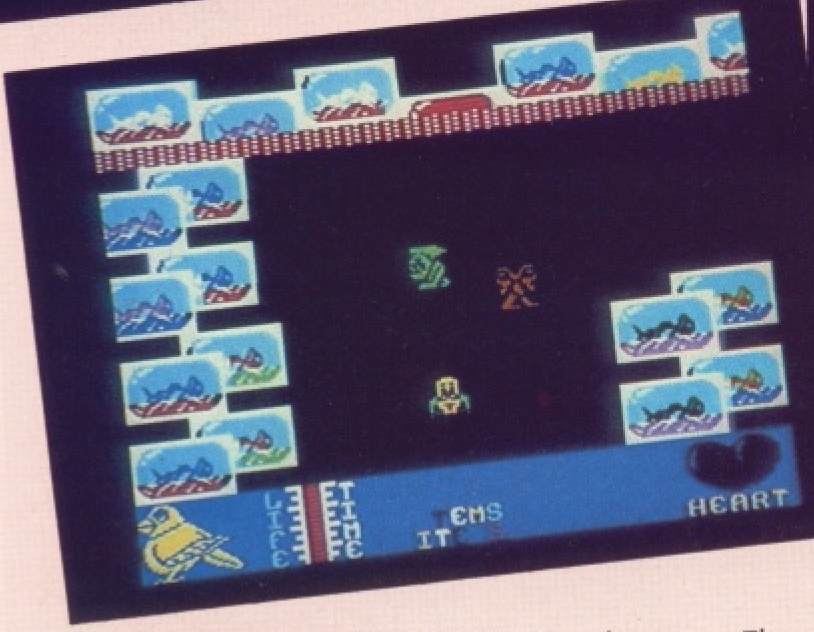
Stuart Cooke goes bonkers about Gremlin Graphics Bounder.

# GAME of the month












EVERY ONCE IN A WHILE A GAME ARrives that you put in your disk drive, play for five minutes, think it isn't very good and put it away. Five minutes later you'll stick it back in the cassette recorder and suddenly find that it's three o'clock in the morning and you've been playing the game for hours. Well, Bounder is one of those games.

Remember the game that you used to play when you were kids (or maybe you still do) where you can't step on any of the lines around the flagstones or the monster will get you? Well, Bounder certainly owes a little of its background to this.

You play the part of a bouncing tennis ball travelling accross a scrolling path. Your way is hindered by a large number of pitfalls. Perhaps the main problem that you have to overcome is the fact that you can only land on the paving stones. If you miss then your tennis ball plummets to the ground a long way below you. I'm not quite sure why the path is floating above ground, but you can't knock the programmer for a lack of artistic licence.

Large gaps in the paving can be bounced across by means of the super bouncers. These are paving stones which have an arrow on them and give your ball that extra power it needs to bridge the gap.

Large walls block your path so you must guide your ball around them. In the meantime men on hangliders and floating logs are trying to stop your progress.

If (or when) you manage to reach the end of a level you'll find the goal awaiting your ball. If you enter this then you'll find your score increasing.

After each level you'll find yourself entering the bonus screen. This consists of a number of paving stones with question marks. Landing on a question mark increases your score. Be warned you only have a limited number of jumps in which to increase your score. Extremely boring but it certainly bumps up your points.

Scattered around each of the levels you will find a number of mystery paving stones. These bare the same mark as those on the bonus level but will not always give you more points. If you are lucky then you will gain extra jumps for use on the bonus level, extra points or extra balls. If however you are unlucky you will find your ball being chomped to death by a mouth or even punctured by a flying dart.

The hazards become more severe the further you get into the game. Missiles are launched from the sea, man (ball?) traps suddenly appear and burst your ball, a plethora of flying beasties get in your way, even some of the paving stones disappear from beneath you. In fact it seems that everything is out to prevent you from reaching the goal and bonus level.

A superb tune sets the atmosphere for your travels along the pathway. The graphics scroll excellently and your ball spins as though it has just left a tennis star's racquet.

As a bonus an extra game, Metabolis, is being given away free with Bounder. Metabolis is a conversion of an earlier Spectrum game and offers very little difference from the original.

You play the part of a little bird who is flying around a vast number of locations attempting to stay away from the numerous baddies. Your aim in life is to collect a number of radioactive pieces scattered around in some extremely inconvenient places.

Mapping the locations for this game is definitely a must as you will soon find yourself lost.

Metabolis offers nothing out of the ordinary and probably wouldn't do too well as a stand alone game. However being included in a package with Bounder makes it an excellent purchase.

Bounder is definitely one of those 'Just another Go' type of games and a must for any serious Commodore 64 collection. Especially when you remember that you are getting another game thrown in for the price of one.

# OR...MADE BY ALIEN



# THE FINAL CARTRIDGE®

## THE FIRST OUTSIDE OPERATING SYSTEM FOR THE CBM 64

**This new operating system built in a cartridge does not use any memory and is always there. Compatible with 98% of all programs.**

Features:

- **DISK TURBO** - 6 times faster disk access - loading and saving.
- **TAPE TURBO** - 10 times faster, even with files - normal Commodore commands - compatible with standard turbo's.
- **ADVANCED CENTRONICS INTERFACE** - compatible with all the well-known centronics printers and Commodore printer programs. Prints all the Commodore graphics and control codes (important for listings).

  Advanced screen-dump facilities. Prints Low-Res, Hi-res and Multicolour. Full page!!! Even from games and paint-programs, like Doodle, Koala Pad etc. Searches automatically for the memory-address of the picture.
- **24K EXTRA RAM FOR BASIC-PROGRAMS AVAILABLE:** Two new commands ,,Memory read'', ,,Memory write''. They move 192 bytes with machinelanguage-speed anywhere in the 64K Ram of the CBM 64. Can be used with strings and variables.
- **BASIC 4.0 COMMANDS** - like Dload, Dsave, Dappend, Catalog, etc.
- **BASIC TOOLKIT** with Auto, Renum (incl. Goto and Gosub), Find, Help, Old, etc.
- **PREPROGRAMMED FUNCTION KEYS:** - Run, Load, Save, Catalog, Disk commands, List (removes all list-protections).
- **KEYBORD EXTRA'S** - Allows you to delete part of a line; stop and continues listings; move cursor to lower lefthand corner. Pokes and Syscalls in Hex. Typ-command operates your printer as a typewriter.
- **COMFORTABLE EXTENDED ML. MONITOR:** - with relocated load scrolling up and down. Bankswitching, etc.
- **RESET SWITCH:** - resets to monitor; resets with old, resets to Hi-Res printing; resets every protected program.

**ON/OFF SWITCH** - we hope you never need that one.

12 Months replacement guarantee.

14 Days money back guarantee if you are dissatisfied.

## SPECIAL INTRODUCTORY PRICE

FOR ONE **£ 50,-**

FOR TWO **£ 37,-** each

FOR THREE OR MORE **£30,-** each

**Don't wait for your friends ORDER ONE NOW!!! Just pay the difference if you reorder within a month.**

H&P COMPUTERS

**U.K. ORDERS** Barclays and Accescards or Cheques should be made out to:
**H & P Computers,(YC) 9 Hornbeamwalk Witham Essex CM8 2 SZ England. Telephone: 0376 - 511471.**

**Part two of Daryl Bowers' machine code arcade game.**



# FRO

IN THIS MONTH'S ARTICLE I am introducing the multi-purpose interrupt handling routine and the end of the routine 'INIT'. We have also got some smooth scrolling and wrap-around! Can you bear the anticipation?

If you look at the source code listing you will see that RESTORE1, RESTORE2 and TABMOV need to be changed, and lines 1540 and 1550 must be replaced with blank remark lines. Now on to heavy stuff!

The first 40 lines are an addition to 'INIT'. The section to set up interrupts is the one with which we are mainly concerned. First, we disable interrupts with SEI. If we did not, and an interrupt occurred while we were changing the interrupt vector, the C64 would probably crash.

Locations $0314 and $0315 hold the two byte address of the interrupt handling routine. This normally points to $EA31 – where the KERNAL takes care of the keyboard input and various other 'events'. We shall replace this with the address of our own routine – 'HANDLE'. This is done in lines 3500 to 3530.

Next we enable Raster Interrupts – in other words, when the raster in the monitor reaches a certain point down the screen an interrupt will occur – lines 3540 to 3560.

Locations $D011 and $D012 hold the 'raster compare value'. If we place a value in these locations it is stored by the Vic chip. When the raster reaches that number of lines down the screen the Vic chip will cause an interrupt. $D012 contains the low byte and bit zero of $D011 contains the high bits of this value. We set up the value of the first interrupt position in lines 3570 - 3610 to $00FF.

Having changed all this we can now enable interrupts again – 'CLI' – and finally we shall turn off the keyboard – lines 3630-3650. We do this because the depression of a key causes an interrupt when we don't want one – try removing

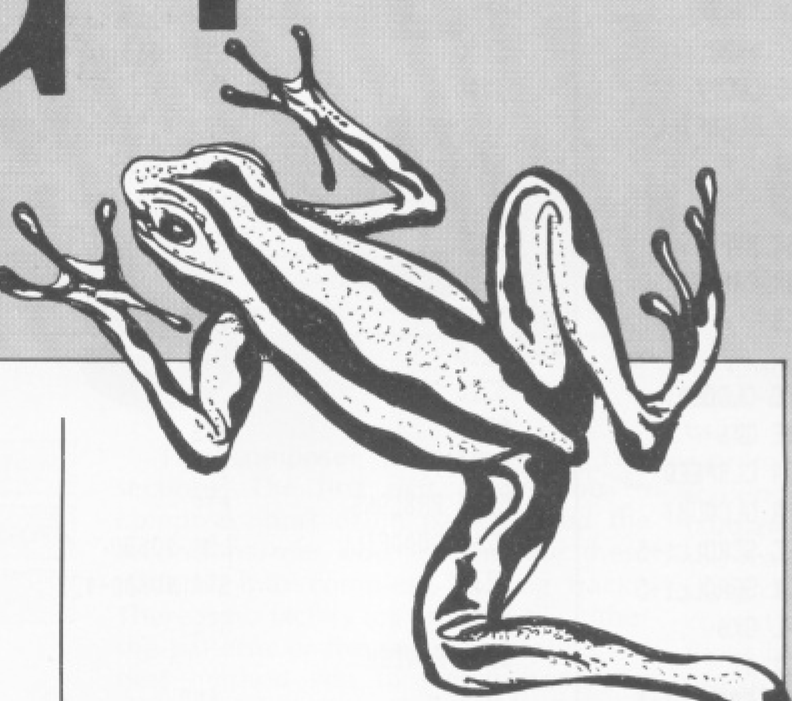
```
330 RESTORE1        .BYTE   2
340 RESTORE2        .BYTE   4

870 STAT1           .BYTE  '    METERS : 00000     LIVES : 3     '
880 STAT2           .BYTE  '    SCORE  : 00000     FOOD  : 99    '

1540                ;
1550                ;

3470                ;SET UP INTERRUPTS
3480                ;
3490                SEI
3500                LDA #HANDLE&255
3510                STA $0314
3520                LDA #HANDLE/256
3530                STA $0315
3540                LDA $D01A
3550                ORA #1
3560                STA $D01A
3570                LDA $D011
3580                AND #127
3590                STA $D011
3600                LDA #$FF
3610                STA $D012
3620                CLI
3630                LDA $DC0E
3640                AND #$FE
3650                STA $DC0E
3660                ;
3670                ;SET SCREEN SIZE
3680                ;
3690                LDA $D016
3700                AND #255-8
3710                STA $D016
3720                ;
3730                ;PRINT STATUS
3740                ;
3750                LDY #39
3760 LOOP18         LDA STAT1,Y
3770                AND #%00111111
3780                STA $0748,Y
3790                LDA STAT2,Y
3800                AND #%00111111
3810                STA $0798,Y
3820                DEY
3830                BPL LOOP18
3840                RTS
3850                ;
3860                ;
3870                ;
3880                ;INTERRUPT HANDLING ROUTINE
3890                ;
3900 HANDLE
3910                LDA $D019
3920                AND #1
3930                BNE RASTER
3940                JMP $EA31
3950 RASTER
3960                STA $D019
3970                LDA $D012
3980                CMP RAST2
3990                BCC POSITION1
4000                CMP RAST2+1
4010                BCC POSITION2
4020                CMP RAST2+2
4030                BCC POSITION3
4040                CMP RAST2+3
4050                BCC POSITION4
4060                CMP RAST2+4
4070                BCC POSITION5
4080                JMP POSITION6
4090                ;
4100                ;
4110 POSITION1      LDX #0
4120                JMP ANYPOS
4130                ;
4140 POSITION2      LDX #1
```

# GGY



```
4150                 JMP ANYPOS
4160                 ;
4170 POSITION3       LDX #2
4180                 JMP ANYPOS
4190                 ;
4200 POSITION4       LDX #3
4210                 JMP ANYPOS
4220                 ;
4230 POSITION5       LDX #4
4240                 JMP ANYPOS
4250                 ;
4260 POSITION6       LDX #5
4270                 ;
4280 ANYPOS          LDA $D016
4290                 AND #248
4300                 ORA SCROLL1,X
4310                 STA $D016
4320                 LDA RAST2,X
4330                 STA $D012
4340                 LDA BORD1,X
4350                 STA $D020
4360                 LDA BACK1,X
4370                 STA $D021
4380                 TXA
4390                 ASL A
4400                 TAX
4410                 LDA TABMOV,X
4420                 STA JUMP+1
4430                 LDA TABMOV+1,X
4440                 STA JUMP+2
4450 JUMP            JSR $FFFF
4460                 ;
4470 FININT          PLA
4480                 TAY
4490                 PLA
4500                 TAX
4510                 PLA
4520                 RTI
4530                 ;
4540 SCROLL1         .BYTE   0,0,0,0,0,0
4550 RAST2           .BYTE   131,167,182,211,251,91
4560 BORD1           .BYTE   0,0,0,0,2,0
4570 BACK1           .BYTE   3,3,$0D,$0C,2,3
4580                 ;
4590                 ;
4600 MOVE1           RTS
4610                 ;
4620 MOVE2           DEC SCROLL1+1
4630                 DEC SCROLL1+2
4640                 LDX SCROLL1+1
4650                 BPL OK2
4660                 LDX #7
4670                 STX SCROLL1+1
4680                 STX SCROLL1+2
4690                 LDY #1
4700 LOOP10
4710                 LDA $05E0,Y
4720                 STA $05DF,Y
4730                 INY
4740                 CPY #$A1
4750                 BNE LOOP10
4760                 JSR PRBLDNG
4770 OK2             RTS
4780                 ;
4790 MOVE4           DEC SCROLL1+3
4800                 DEC SCROLL1+3
4810                 LDX SCROLL1+3
4820                 BPL OK4
4830                 LDX #7
4840                 STX SCROLL1+3
4850                 LDY #1
4860 LOOP7           LDA $06D0,Y
```

these lines and see what happens!

Lines 3690 to 3710 reduce the number of columns of characters on the screen to 38, by blanking out the first and last. This means that characters will smoothly scroll off the edges of the screen; again, try omitting them and watch the left hand side.

The end of this routine simply places the data in lines 870 and 880 on to the screen. Note the 'AND' statements. These are used because the assembler converts alphanumerics in BYTE statements into their ASCII values. The C64 screen, however, uses values 64 less than the ASCII equivalent. These 'AND's, therefore, remove bits seven and six from the values – the equivalent of '-64'.

## Handling Interrupts

The routine 'HANDLE', is designed to be totally portable, that is, you can use it in any program. Five tables are used:

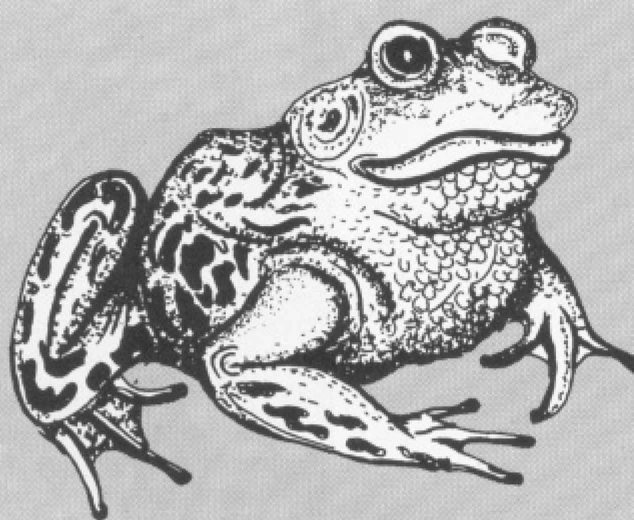
4540 SCROLL1 : this contains the X smooth scroll value.
4550 RAST2 : the Y positions where raster interrupts occur.
4560 BORD1 : the border colour.
4570 BACK1: the background colour.
1430 TABMOV : the address of the scroll routine.

I have allowed six interrupt positions – you can add more or use less – and it is easily possible to add more tables defining further effects; Y smooth scroll for instance.

Now we know that if any interrupt occurs the processor will complete what it is doing and jump to 'HANDLE'. The first thing we must decide is whether the interrupt has been caused by our raster compare value or by some other source. This is done in lines 3910 to 3950. When a raster interrupt occurs, bit zero of $D019 is set to one. If this is not set we jump to $EA31 – the normal KERNAL routine. A 1 must be written back into $D019 to clear the register, ready for the next interrupt (a rather strange way to do it if you ask me!) in line 3960.

The next dozen lines check the value in $D012 (the current raster position) against our table of values, and branches to

```
4870            STA $06CF,Y
4880            LDA $DAD0,Y
4890            STA $DACF,Y
4900            INY
4910            CPY #$28
4920            BNE LOOP7
4930            JSR ROADFILL
4940 OK4        RTS
4950            ;
4960 MOVE5      JSR MVFROG
4970            JSR PRFROG
4980            RTS
4990            ;
5000 MOVE6      DEC CLCOUNT
5010            BNE OK6
5020            LDA CLSPEED
5030            STA CLCOUNT
5040            DEC SCROLL1+5
5050            LDX SCROLL1+5
5060            BPL OK6
5070            LDX #7
5080            STX SCROLL1+5
5090            LDY #1
5100            LDX 1064
5110 LOOP9
5120            LDA 1064,Y
5130            STA 1063,Y
5140            INY
5150            CPY #121
5160            BNE LOOP9
5170            LDY 1104+39
5180            STY 1104+79
5190            LDY 1064+39
5200            STY 1064+79
5210            STX 1064+39
5220 OK6
5230            RTS
5240            ;
5250 PRBLDNG    RTS
5260 ROADFILL   LDA $06D0
5270            STA $06D0+$27
5280            RTS
5290 FINISH
5300            .END
```



the appropriate 'POSITION'. At this point the value of X is set up ready to index into the tables.

The routine ANYPOS is used for all positions, and sets up the X smooth scroll, next interrupt position, background and border colours and the address of the movement routine. The effect of this is to allow 'bands' of characters on the screen to scroll at different speeds, in different directions and with different colours. No mean feat!

'FININT' simply restores the registers to their correct values and returns from the interrupt.

## The Scroll Routines

Essentially, all the scroll routines from line 4620 to 5240 are the same, so I will concentrate on just one of them: 'MOVE4'.

The 'band' in area four (the road) will be moving from right to left at a rate of two pixels every interrupt (60th of a second). This is achieved in lines 4790 to 4800. If it were to move right we would INCrement the value in 'SCROLL1+3'.

Next we check to see if we have scrolled a whole character – lines 4810 and 4820. If the value has reached minus one we replace it with seven (lines 4830 and 4840) and procede to scroll the characters eight pixels (one character) to the left – lines 4850 to 4920.

'ROADFILL' is the routine which fills in the right-most character, in this case with the character which 'falls' off the left.

It will be seen that 'MVFROG' and 'PRFROG' have now been moved to the interrupt routine. This is to ensure that there is no flicker when the sprites are moved, since they are printed when the raster is below them.

## There's More

When typing in the listing, change the symbols '&' and '/' to '<' and '>' for example:

LDA # HANDLE & 255 becomes
LDA # <HANDLE

This is because my assembler insists on being awkward when it does a listing! 'START' should now be at location $2335 (9013)
'FINISH' should now be at location $2526 (9670)

Next month we shall introduce the frog movement routines, and a joystick reading routine.

# THE • BEAT • GOES • ON •

**Syntron's Digidrum is foot-tapping, hand-clapping good, according to Eric Doyle.**

THE HEART OF A GOOD BAND IS THE ability of the rhythm section to mark time with the accuracy of a metronome and it is not surprising that microchip technology has developed the rhythm generator as the ultimate musical timepiece. The problem is that, unless you are willing to pay a small fortune, the current generation of machines has as much soul as the ticking of a clock. But this appears to be changing.

Syntron's Digidrum is a flexible drumkit synthesiser which allows for those little touches of individuality which normally differentiate the human from the machine. There is sufficient programming flexibility to allow changes in rhythm and variation in style and to convince the average listener that he is listening to a syndrum kit played by a human.

The Digidrum package consists of a set of disks, containing the composer software and instrument databases, and a cartridge which plugs into the user port of the C64. The cartridge has a standard jack output for connection to an external amplifier and a trigger output which can be used to keep several slave sequenced synthesisers in time to the beat.

On loading the software, you already have a standard drum kit of seven instruments: base drum, snaredrum, three tom toms (large, small and floor), a crash cymbal and a hihat. These give a range of eight sounds in all because the hihat cymals are classed as two separate instruments whether open or closed. The sample programs give an opportunity to hear how the drumkit sounds and very impressive it is too, despite the slightly electronic sound.

The composer program is in two sections. The first part allows you to compose short drum patterns and the second permits you to combine these patterns into complete backing tracks. There is no facility for printing out either the patterns or the songs so I found the best method was to switch from one section to the next and assemble the song from each pattern whilst the sound was fresh in my mind.

The pattern composer will permit 51 different patterns with a length of 38 beats. This is an arbitrary figure because the tempo can be changed over 64 steps which gives a range from the very, very slow to the impossibly fast. The instructions suggest that the mean value is around 44. The screen display looks like a familiar musical stave but there are eight lines instead of the normal five, each with a letter corresponding to one of the instruments.

Tunes are entered by moving the arrow cursor along a tessalated vernier scale at the bottom of the screen until the correct position is reached. Then the letter key corresponding to the chosen instrument is pressed. The result is that a symbol appears on the relevant line and the instrument sound is heard. After a few notes have been entered, you can listen to the pattern simply by pressing a function key.

The number of instruments which can be sounded at the same beat position is limited to three. A drummer only has two hands so the program makes it impossible for a snaredrum, tom tom and cymbal to be sounded at the same time but a snaredrum, cymbal and bass drum can. The flexibility of this system means that a respectable and plausible drum solo can be created using triols, syncopation or any other device or basic rhythm which takes your fancy.

Each pattern does not have to be a full 38 beats long it can be foreshortened by placing an end bar at any position along its length.

Once a few patterns have been created they can be combined into a song using the second program. This is extremely simple to understand. The screen shows several columns which are tabulated in rows from one to 100. This is the maximum number of steps which a song can have but since each step can consist of the same pattern repeated 100 times it doesn't take an Einstein to work out that there is room enough for even the most ambitious project.

Entry of a song is made by selecting the pattern number and the number of repeats which are then displayed on the song screen. The tune can then be played in full or part to see how it sounds and if necessary a pattern can be called up and trimmed until the fusion is complete.

As in a word processor, there are several keys which allow the selection of blocks of the song which can be copied, deleted or inserted. Similarly patterns can be copied from one pattern position to another so that small changes can be made to create a new variation to add interest to the generated rhythm.

There is room for 10 songs which use the same bank of patterns and these can be saved to disk for recall when necessary.

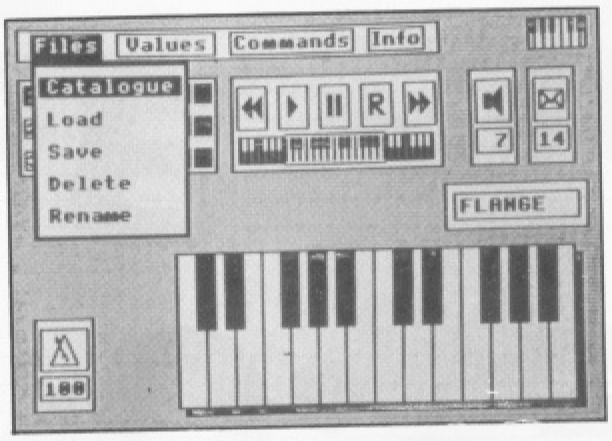
In addition to the basic drum kit there is the option to replace any or all of the instruments with new ones which range from the hi-tech syndrum sounds to the more unusual percussion instruments such as a metal bar or even a very realistic hand clap. The limitation is still eight sounds and only three to a beat.

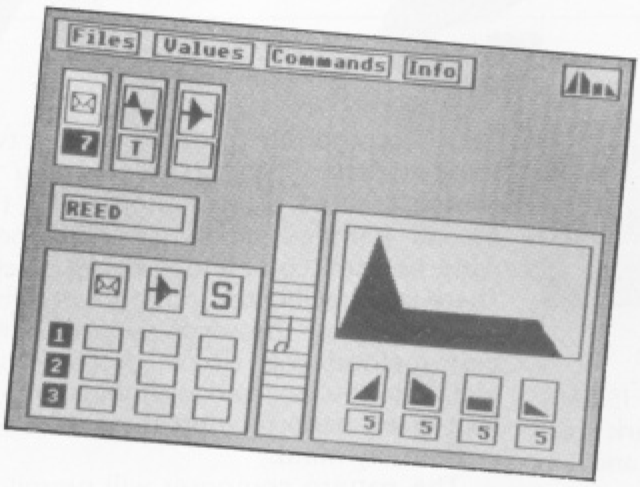
In the studio it would be ideal for making demo tapes and the only complaint I have is the length of time taken to create the patterns. I did find this stage quite enjoyable, however, giving plenty of freedom to experiment. For live performance, Digidrum would only be practical if all the music used the same percussion set or the performance was organised to allow time for loading.

Although there are limitations to live performances, I am quite sure that it won't be long before the strains of Sid Syntron's scintillating syncopation are heard in the local pub.

# NOW

**Ian Waugh has been discovering Island Logic's The Music System.**




THERE ARE ALREADY DOZENS OF music packages available for the C64, all with their various strengths and weaknesses. Anyone who launches yet another package must think they have something pretty special.

Even if your ears turn blue at the mere mention of a BBC computer, you may well have heard about Island Logic's *The Music System* which was hailed by music-minded users as the best thing since memory expansion boards. The development team, System Software, has now produced a version of *The Music System* for the C64 and 128 and distribution has switched to Firebird.

As you might expect, the superiority of the SID chip immediately gives any Commodore music program a tremendous advantage over a simiar one on the Beeb. Not content with this, System Software has included a MIDI module which allows access to external synthesisers.

*The Music System* (or TMS as it is usually referred to) was highly regarded not only for its music features but also for its use of icons and pull-down menus. The Commodore version can only enhance System's programming and design reputation as its use of these features are not only superbly implemented but they also make its operation relatively simple. And with six modules to choose from and over a hundred functions available from the keyboard, it needs to be simple. Most keys perform the same functions in each module so it's nowhere as near so daunting a task as it may at first appear. A handy Quick Key Guide helps enormously and you'll find after a little use that the keys fall under your fingers quite naturally.

Right! Let's take a look at the modules. These are the Editor, Keyboard, Synthesiser, MIDI, Printer and Linker whose icons are displayed on the main menu screen. Each module has a Command Line running across the top of the screen from which the pull-down windows. . .er. . .pull down. There are four menus here: Files, Values, Commands and Info and each is selected by pressing one of the function keys. The information given in the Command Line menus differs from module to module but is similar in type.

Files controls the loading, saving, renaming and deletion of files and only those relevant to the module you are in can be accessed from that module. Values holds such information as key signature, tempo, octave, volume and voice number. Commands is generally concerned with instructions which affect the whole or large portions of the piece such as delete track and clear all tracks. It is also home for a set of macro commands such as settting markers, copying sections to the notepad (more of that in a moment), adjusting barlines, swapping and copying envelopes and filters, etc. Info displays general information about the state of your composition, for example note storage space and the names of current music and sound files.

Moving on to specific modules, the one you are likely to use the most is the Editor. This displays a treble and bass clef in what is referred to as the Voice Monitor (VMW). Notes are entered here. The VMW only shows one voice at a time but you can flip from one to another at the press of a key and the bars are always aligned.

Notes are moved up and down the staves to select pitch and the note name and octave is displayed in a small box on the top right of the screen. Each note can be assigned a different volume level and any one of 15 envelopes. The current bar number is shown and horizontal bars called barmeters indicate how much has been recorded on each voice. You can scroll through the score with ease and notes can be inserted and deleted at any point. A full range of accidentals can be used including double sharps and flats for the musical intelligentsia and notes can be turned into triplets and tied although no more than two notes can be tied together at once.

The program will insert bar lines automatically if required and you can insert first and second time bars. Another feature of TMS is the ability to define loop sections. When played as part of a tune, a loop section will keep repeating until the whole tune has finished. Each voice can contain up to 20 different loop sections so you can quickly select one of a number of repeating bass or rhythm patterns to improvise over or for use in a tune.

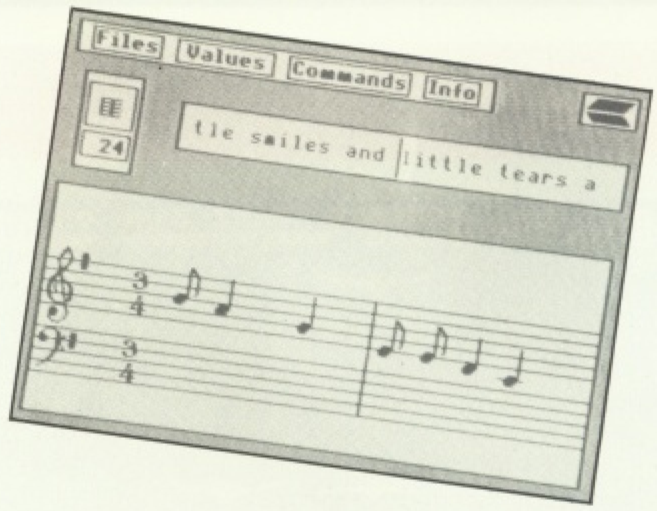
If all these features have your mouth watering and fingers itching it's only fair to warn you that we're only up to page 25 of the manual. There is lots more to come.

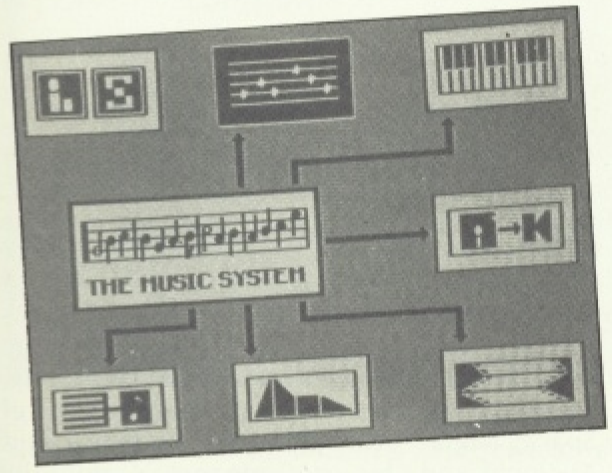
From the Commands menu you can call up a set of macro commands. These operate upon a section of a voice which has previously been marked with two markers. Macro commands include transposition and envelope and volume assignment.

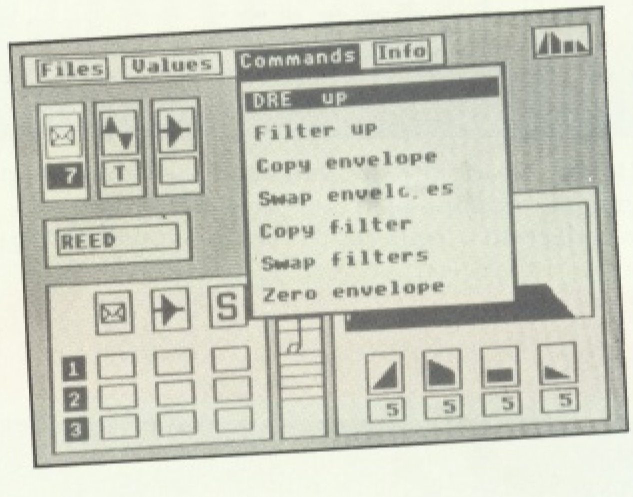
Yet another feature is the Notepad. This is used to store a section of a voice which can be moved to another part of the same voice or a different one. It can also be used to merge two music files together and notepad files can be saved and loaded like any other although only one can be held in memory at a time.

If you prefer to tap out tunes on the Commodore's keys, enter the Keyboard module. Real-time note entry from QWERTY keys is not the easiest way of writing a tune but the Keyboard module

# HEAR THIS




helps to make a difficult job as easy as it can be. Three voices are entered monophonically with horizontal bar-meters to show you how much room is available for each voice and the VMW can be called to display the notes. The Keyboard is entered in practice mode, or Tinkle as it is called in the manual. Music and Sound files can be loaded although only music files can be saved from here. The manual is kind enough to explain that anything but simple tunes may produce untidy compositions – and it's right – but then the Editor module includes a Block Tidy command to help put right the mess you make.

The Synthesiser module is where you come to grips with SID. It must be the most complete and sophisticated SID chip editor yet devised. Graphic displays give a visual indication of SID's parameters and you can load and play a music file to hear the effect of the envelope as you alter it on a proper tune. The envelope includes waveform and filter selection and you can name each creation for easy reference.

The Synthesiser includes extra facilities like sweeping the pulse width, pitch and filter without using another voice – an extra bit of magic.

The MIDI module will be attractive to a lot of users but let us not forget that a MIDI interface is also required. The program supports the SIEL (£79) and the Passport (£110) interfaces and although it may work with others this is not guaranteed. Although the concept of MIDI was to produce a standard set of digital information signals, the standards haven't yet filtered down to MIDI interface manufacturers.

This module is basically a six-track polyphonic real-time sequencer. It records most performance information but not pitch-bend which cuts off any following data. I wonder how this got past the debugging team. You can set the tempo, select and delete individual tracks and pause recording by pressing the space bar. There are no channel assignment or editing facilities but it is a rather excellent extra and TMS is not, after all, a dedicated MIDI program. A big plus is the ability to convert MIDI files to music files playable by SID and vice versa. Regrettable but rather obviously, any mutli-part polyphonic pieces are converted into three monophonic lines. Performance data goes but voice numbers in the range one to 15 are converted to envelope numbers. You can use the MIDI module for real-time input and tidy up the parts later in the Editor.

The Printer module supports Epsom and Commodore printers and you can add lyrics to the score, too, providing a convenient copy of your masterpiece.

Finally, the Linker module is used to link individual music files. It is the only way tempo, key and time signature changes can be implemented in a single piece and, of course, it allows mammoth compositions to play through in total. Up to 26 files can be loaded, memory permitting, and arranged to play back in a sequence which can contain up to 99 items. The whole arrangement can then be saved as one file for convenient reloading and playing although you can't play this back through the MIDI module.

The 94 page manual is well-produced, well-written, easy to read, full of illustrations and a comprehensive index will lead you to virtually every occurrence of every aspect of TMS. Umpteen demo tunes are supplied on the disk with even more on the other side (a bit naughty, these reversible disks, aren't they?).

You may have guessed by now, but if you haven't, here it is: TMS is the ultimate Commodore music editor for the SID chip. The MIDI module is a bonus although musicians with serious MIDI requirements will need a dedicated software package. TMS is easy to use in spite of its wealth of features and it's fun. If you are at all interested in making music with your Commodore, I can not recommend it too highly.

The Advanced version of TMS containing all the modules described above retails for £39.95 and is available only on disk. A smaller version containing only the Editor, Keyboard and Synthesiser modules sells for £17.95 on disk and £14.95 on cassette. What will System produce for the Amiga?

The Music System is available from: Firebird Software, Wellington House, Upper St Martin's Lane, London, WC2H 9DL.

Originality | Playability | Graphics | Value For Money

## Yak's Progress

**Llamasoft £11.50 disk £9.45 cassette C64 + Joystick**

Originality 10 Playability 8 Graphics 8 Value For Money 10




JEFF MINTER HAS BEEN around since Vic was just a lad and now a collection of Minter games is available under the title of Yak's progress.

As a document of one man's fight to tame a machine, this compendium is fascinating and allows the newer C64 owners to catch up on some of the best and most unusual zapping games to be devised for the machine.

Eight games for the price of one is an offer that few will refuse and for many it will provide the chance to get turbo versions of games already in their collections as well as filling up any gaps.

Attack of the Mutant Camels and Revenge are both here alongside Matrix, Laser Zone, Sheep in Space, Metagalactic Llamas, Ancipital and Hover Bovver.

There are enough camels to give you the hump, sheep to drive you baa-rmy and llamas to drive you bananas. The graphics vary as Minter's programminng skills develop and the accompanying booklet gives the lowdown on the workings of the programmer's fevered brain.

To try to describe the games would probably take most of the review space in this issue because the games are unlike anything to be seen elsewhere. Imaginative and demanding, they are not everyone's cup of tea but the Minter following is by no means a minority movement.

**E.D.**

## The Last V8

**Mastertronic Mad Series £2.99 C64**

Originality 6 Playability 8 Graphics 7 Value For Money 9




"V8 RETURN TO BASE immediately" crackles the voice in your headset and you're off in a race against time as you try to get home before a delayed attack nucleur warhead wipes you off the face of the earth.

As a scientist working on project V8, you were hidden deep underground at the time of the holocaust. You have now been allowed to try and make contact with the survivors.

The screen is divided into two windows. The bottom depicts your instrument panel whilst the top gives a birds-eye view of you and your surroundings. The dials you have to watch are fairly simple – they include speed, distance to base and time before the explosion. You will, however, have to keep your eyes firmly on the road.

Providing you stay on the road, all well and good but you soon discover that you cannot take corners at 410kph. One crash and that's it – game over.

If you slow down to take corners, there is no time for you to return to base before the bomb detonates. Eventually, I realised that some of the trees I kept hitting could, with slightly better driving, be avoided

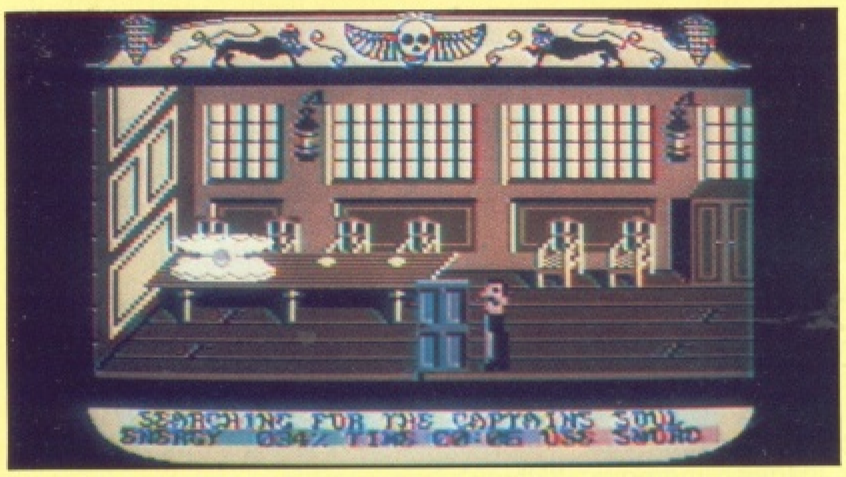
Strangely, the speech adds little to the game's atmosphere but the rest is extremely addictive. The car is very responsive and the music is great. You don't have to be a racing fan to enjoy this one.

**G.H.**

## Blackwyche

**Ultimate £9.95 C64 + Joystick**

Originality 6 Playability 7 Graphics 7 Value For Money 7



MUCH HAS ALREADY BEEN said about the C64 answer to the Marie Celeste, the bad ship Blackwyche. In my opinion a lot of this was pure hype. It is an adventure style game and has little to commend it over many similar offerings.

Starting on the upper decks you are immediately harried by winged demons and the occasional flying octopus! Unarmed and helpless, your first task is to examine the cabins to find a sword guarded by two nasty skeletons. Attacking from both sides, they beat you with bony fingers, knocking down your energy level as they do so. You must find dubloons to replete your power.

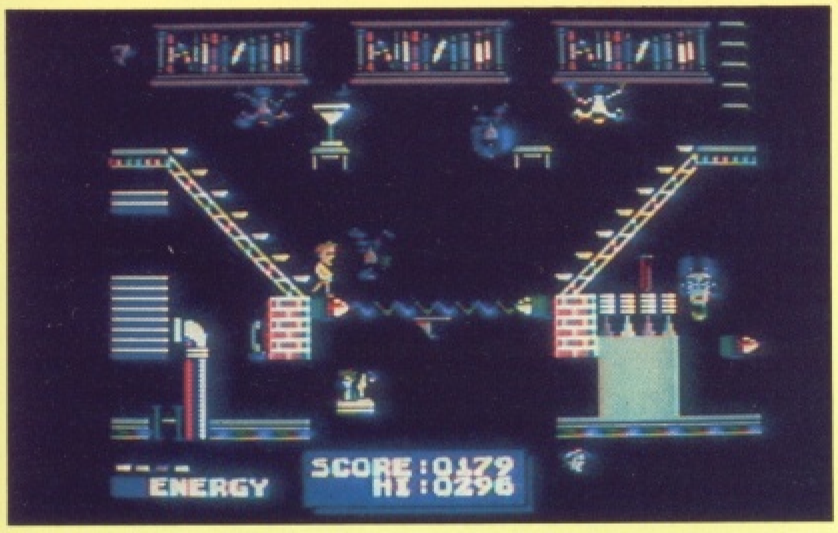
Once you have the sword you can attack the flying menaces outside but it is ineffective against most of the ghouls in the other cabins. Occasionally you will get a surprise as you enter a cabin, the floor gives way and you end up dazed on a lower deck. Ladders lead up and in this way you can tour the whole ship making discoveries but no friends.

The locations are numerous but the graphics are very repetitive. Only colours differentiate which deck you are on. There is some suitably nautical music over the opening screen but during play there are only the sound effects which are fairly unimaginative.

**E.D.**

## Dynamite Dan
**Mirrorsoft £7.95 C64**

Originality 5 · Playability 7 · Graphics 7 · Value For Money 6




ARRIVING BY ZEPPELIN AT the hideout of Doctor Blitzen is not the subtlest approach Dynamite Dan could have made. With the assistance of his henchwoman, Donna, the doctor has set into operation a multitude of defence systems and it is up to nimble-footed Dan to avoid these obstacles, steal the plans for the super psychron mega-ray which Donna and Blitzen have secreted in the safe, and make his escape.

This is a new platform game from Mirrorsoft and it is fiendishly difficult. As you guide Dan around the house looking for the eight sticks of dynamite needed to blow the safe, all manner of creatures have to be avoided while keeping an eye on your energy level. Food is found at regular intervals which will boost both Dan's energy and your score depending on the type of food found.

There are other objects which score points but the most valuable are the test tubes. The score for this discovery is a miserable 25 points but they do give Dan an extra life which is absolutely essential for success.

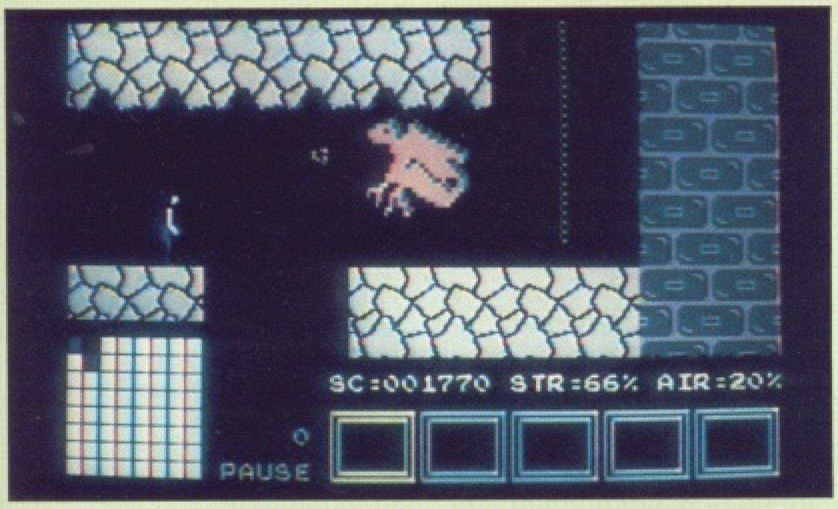
At the bottom of the house is a river and Dan displays what is quaintly referred to as 'negative buoyancy' – he can't swim. This is the most dangerous of all the game's elements because one dip in the water and all Dan's lives are lost.

The game will challenge any platform fanatic with a yearning for explosive fun.

**E.D.**

## Hero of the Golden Talisman
**Mastertronic Mad Range £2.99 C64**

Originality 6 · Playability 4 · Graphics 4 · Value For Money 5




MISSING MY FIRST JUMP IN Golden Talisman, I was more than a trifle surprised to find myself floundering under water being chased by a shoal of ferocious looking piranha. As usual, I had not read the instructions, otherwise I would have realised that a fair proportion of this arcade adventure is of a subaquatic nature.

In order to find the five pieces of the talisman, you must manipulate candles, keys, ropes and spells as you make your way through the various tunnels. Assorted monsters including particularly vicious fire breathing dragons are intent on stopping you. You can carry up to five items at once, the lefthand box showing the one currently in use. Also shown on the screen is a stylised map of where you have been, indicators of your strength remaining, oxygen carried and a large map of immediate surroundings. Your strength and oxygen supplies can be increased by collecting fruit and bags of air that appear periodically.

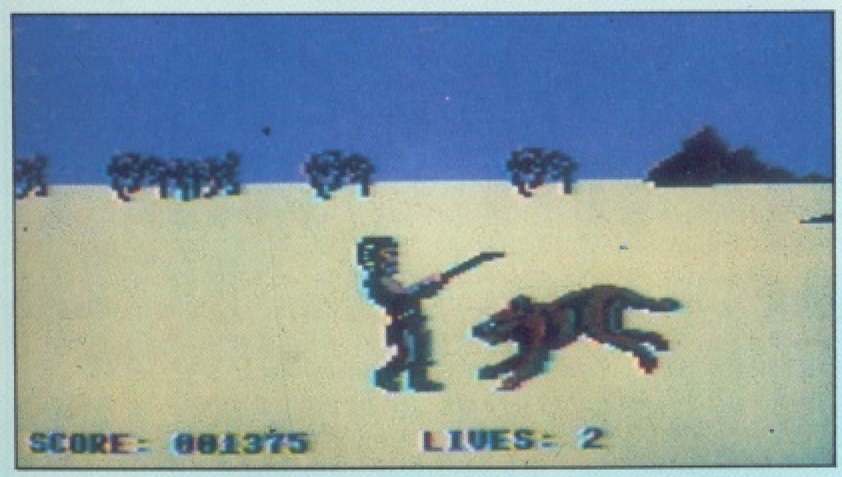
Movement is straightforward, the only tricky bits being the jumps which involve the diagonals on the joystick. There is no scrolling between them so you are never sure what you will find on the next one. The graphics are large and blocky with some of the movement being jerky.

This game has some original ideas but they don't quite get together and the overall impression is uninspiring.

**G.H.**

## Fighting Warrior
**Melbourne House £9.95 cassette £14.95 disk C64**

Originality 3 · Playability 5 · Graphics 6 · Value For Money 5




AS PRINCESSES EVERYWHERE are wont to do, yours has managed to go and get herself kidnapped and you set off to rescue her. The setting is ancient Egypt and in order to achieve your quest, you have to battle against assorted creatures such as humans with jackal heads and winged demons.

Both you and your opponent try to hack each other to bits with swords. The amount of damage you can sustain is depicted by a number of arrows at the bottom of the screen and a successful strike reduces this total by one. When it reaches zero, your adversary dies or you lose one of your five lives.

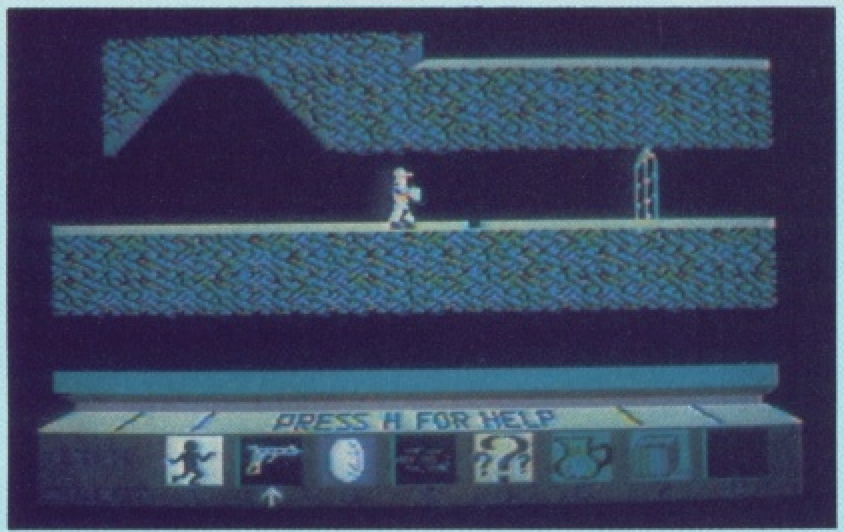
Combat itself gives you a choice of three aggressive and four defensive manoeuvres. You can aim a high, medium or low blow and can jump, duck and move forwards and backwards. In practice, the battles tend to degenerate into a slogging match with both sides standing still and trading blows. After the combat, you get the chance to take a swipe at the magic vase that appears. This may increase or decrease your strength, take you to another zone or force a battle with a god.

Fighting Warrior lacks any lasting appeal. Every battle is much the same as the last and there is only a limited range of movements available.

There are better combat games on the market. **G.H.**

## Journey
**CRL £8.95 C64 + Joystick**

Originality 5 | Playability 5 | Graphics 3 | Value For Money 3




KEEP YOUR HANDS FREE AND hold on to the sides of the vertical shafts as you climb down otherwise you could come to a disastrous end.

The adventurer must collect 11 treasures from the centre of the earth and bring them to the surface. Dragons, magnetic fields, dynamite, detonators etc. hinder your search for the treasures. The player can escape from the vampires by climbing up or down a shaft. There are guns which can be used to kill deadly vampire bats and oxygen pills which can be taken when you enter a gas chamber. To pick something up you must stand over it and push up on the joystick. If you get too heavy you can drop something by pushing down on the joystick.

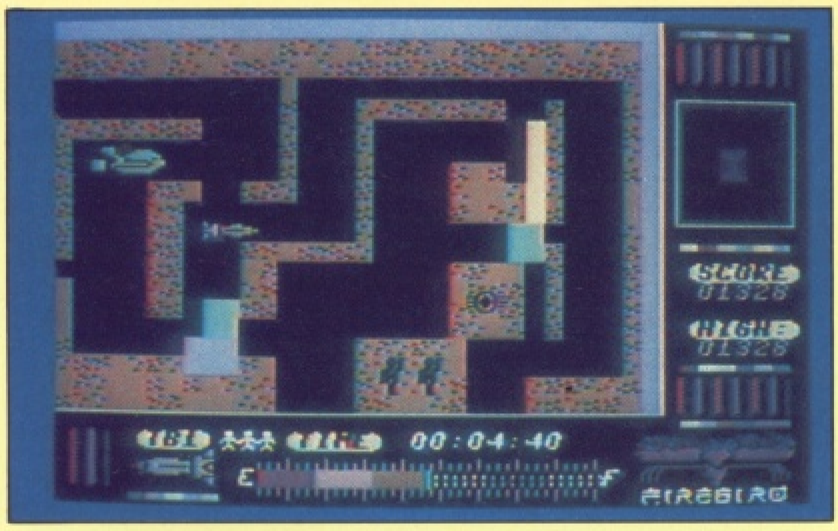
This is yet another arcade strategy game of the Dungeons and Dragons ilk. The special effects show you somersaulting to your death and those used in the transporters are original and good. There are two speed levels, fast and normal. The documentation is adequate and full playing instructions can be found on the back of the cover. The player has three lives and starts each time at the surface. There are different levels of skill the first one being that of a brainless snail.

At first I was continually having to start over again after somersaulting to my death. I found the trick is to hold on to the sides as you climb down. I recommend this challenging game to anyone who wants a few hours of fun.

**J.L.**

## Thunderbirds
**Firebird - Super Silver Range £3.95**

Originality 8 | Playability 8 | Graphics 9 | Value For Money 5




A TEAM OF ARCHAEOLOGISTS have got themselves trapped in an Egyptian tomb. They send out a mayday distress signal which is picked up by International Rescue. Thunderbirds are go.

You control both Thunderbirds 1 and 2 (the fire button toggles control between them) and you have to find your way through a maze of passageways inside the tomb. As in all the best tombs though, it contains a huge number of traps. These come in two forms – huge stones that block the passageways and guardians such as mummies and spiders that try to stop you.

The main problem is the blocks. These are of three types. TB1 can only move blue blocks, TB2 green blocks and both can shift yellow blocks. The maze has been very ingeniously designed and it requires considerable planning to get through. Frequently, you think you have solved a problem only to find that one of your craft is blocked in. To get to the correct position to shift certain blocks, one of the TBs is quite likely to have to detour through three or four other chambers first.

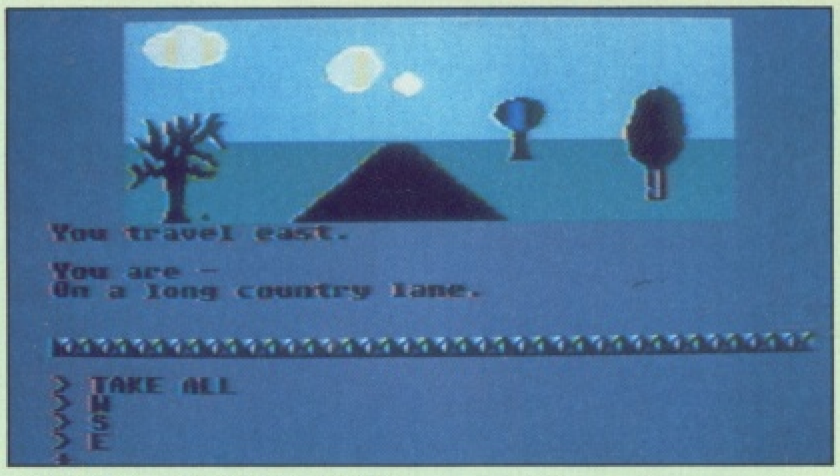
TB2 can carry certain equipment and a menu lets you choose what to take. These items may help you in a later stage of the game but you have to find out how and where. Extra fuel is a must.

Thunderbirds, although not graphically brilliant, is an excellent game which is guaranteed to keep your grey cells ticking over. FAB Virgil.

**G.R.H.**

## Quest for the Holy Grail
**Mastertronic £1.99 C64**

Originality 1 | Playability 1 | Graphics 4 | Value For Money 5



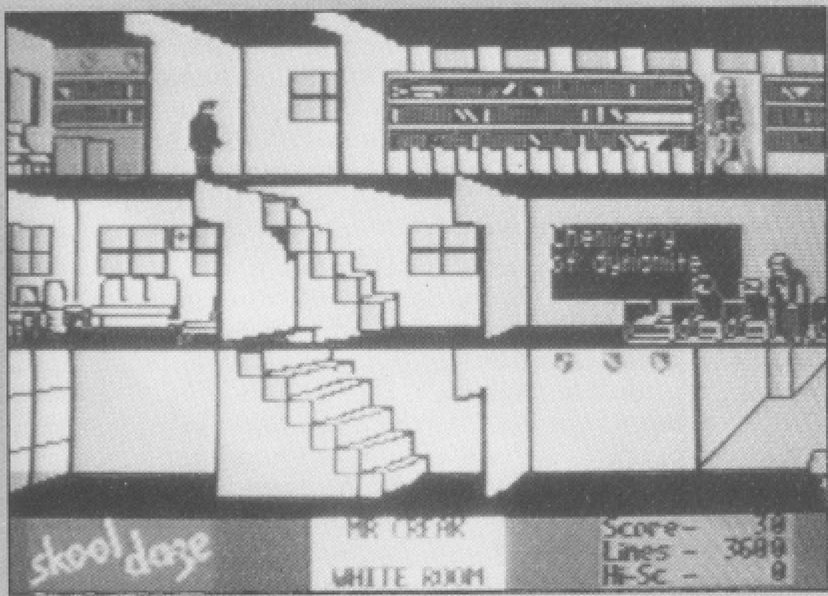

"MONTY PYTHON WITH chips" reads the inlay. This supposedly wacky adventure game should be served with large dollops of green slime. Bearing a striking resemblance to Scott Adams' original adventure games of yesteryear, this game tries to improve on them by using graphics, but not to any success.

Sir Tappin is on a quest to find the Holy Grail. The game starts off in a Forge where the player meets a CND nut with a nuclear powered lamp and a key. He is able to go west, east, north, south, up or down. When Sir Tappin has been squashed by a falling tree or drowned by flying snot or has befallen some other tragedy he restarts his quest in the forge again.

This value for money game offers the enthusiast on a tight budget many hours of exasperating fun and a large vocabulary of four letter words. The writers have left all clues, instructions and commands of how to play to the players imagination or previous experience. Using 'Help' yields "Yes, you'll need it", "No, I'm sulking" or just plain "No!". There are a few original results for some commands – try 'quit' for instance or use 'drop' and suffer the consequences once you are carrying the nucleur powered lamp! I found the language rude andpuerile,and do not recommend it to anyone who wants a challenging and exciting game to play. **J.L.**

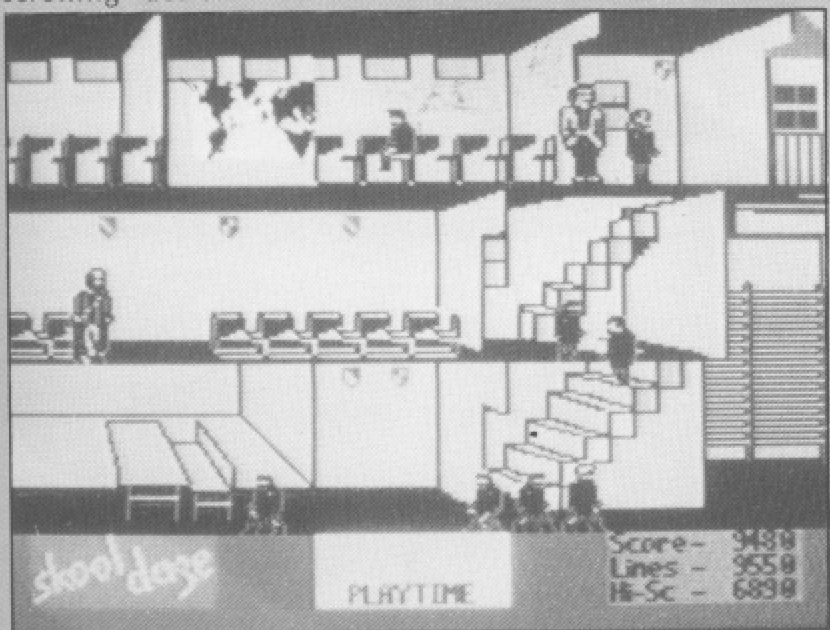
## Skool Daze
**Microsphere £6.95 C64**

Originality 9 Playability 8 Graphics 7 Value For Money 8




CONVERSIONS FROM SPECtrum to C64 often disappoint me but the plot of Skool Daze is so good that even the jerky scrolling action does not detract from the enjoyment.

It tells a tale with which everyone is familiar, the panic just before the school reports are issued. And it is a matter of life and death to steal your report before the Headmaster sees it.




The hero is called Eric, a good name for a hero if ever I heard one, and he must obtain the secret combination of the schoolmasters' safe if he is to save his reputation. If the boy's name does not appeal to you, the names of the main characters can be changed at the start of play.

Like all schoolboys, Eric has lessons to attend and if he is caught wandering about during lesson time, or messing about generally, he will be given lines by passing teachers. If he is clever he will make sure that someone else is closer to the teacher when misbehaving because the blame generally falls on the nearest boy. If Eric amasses more than 10,000 lines he is immediately expelled and the game starts again.

To find the combination, Eric must jump up and hit all of the shields which are hanging on the walls around the school. Some of the shields are too high for a mere schoolboy to reach so violent measures are called for in desperate circumstances. This may involve knocking down a fellow pupil or a schoolmaster and using them to give the necessary height to reach the shield.

When all of the shields have been set in motion, Eric must knock down each teacher in turn to reveal a single letter of the combination. Unfortunately, the dithering old History teacher can only remember his if he sees his own date of birth written on the blackboard. Clues are given to the date during the game but if you guess wrong the nasty little swot of the class will split on you for writing on the board.

Once Eric knows the combination he must rearrange the letters into the correct order, he only knows that the headmaster's come first. The guess is written on a clean blackboard and then he must rush to the staffroom, jump up in front of the safe but if it doesn't open he must go and guess again.

Recovery of the report does not end the game because the shields must be hit again to stop them flashing. Skillful use of a catapult can always help with hitting the shields but this does run the danger of getting lines.

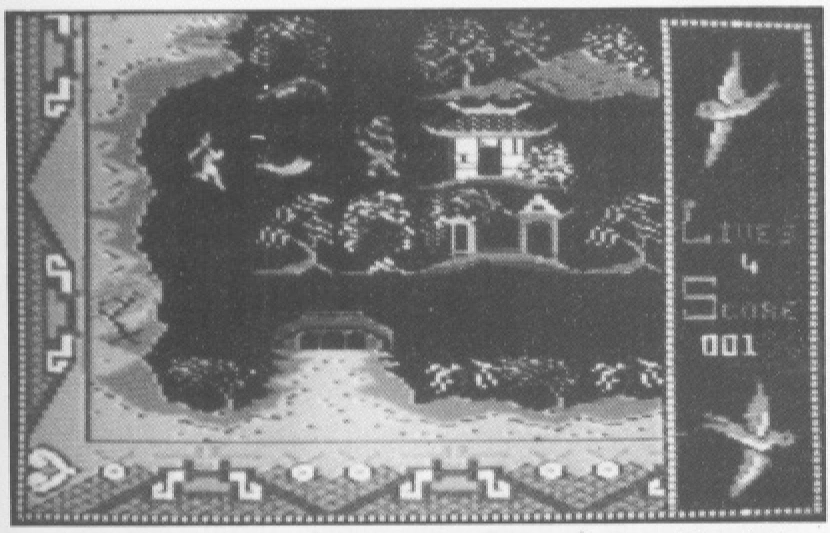
The screen is a hive of activity but it's a pity that more was not made of the 64's capabilities. A touch of blatant sexism creeps in with all of the characters being male but in the tradition of Tom Brown's Schooldays and Billy Bunter I shall overlook this and not put Microsphere in detention.

The old saw about these being the happiest days of your life is laid bare by this, the most traumatic, game of your life.

**E.D.**

## Willow Pattern
**Firebird £3.95**

Originality 6 Playability 7 Graphics 9 Value For Money 8




MOST PEOPLE WILL AT SOME stage have eaten a meal off plates with the willow pattern design on them. The design represents an ancient Chinese fairy tale and Firebird's latest game is an arcade adventure based on that story.

You play a mere clerk who is in love with a princess. The trouble is, she is promised to a merchant so you decide to break into her palace and rescue her. To do that, you must find your way through a maze, find certain objects, overcome the palace guards and then escape to a boat.

In order to defeat a guard, you must throw a sword at him before he throws one at you. You can only carry one sword at a time and so you will have to backtrack a lot. Extra swords can be found lying about or you can entice a guard to throw one and then dodge out of the way.

Occasionally, you have to cross a bridge which you do by leaping from stepping stone to stone. It's not quite that easy as three giants try their hardest to knock you over and so timing your leaps becomes crucial.

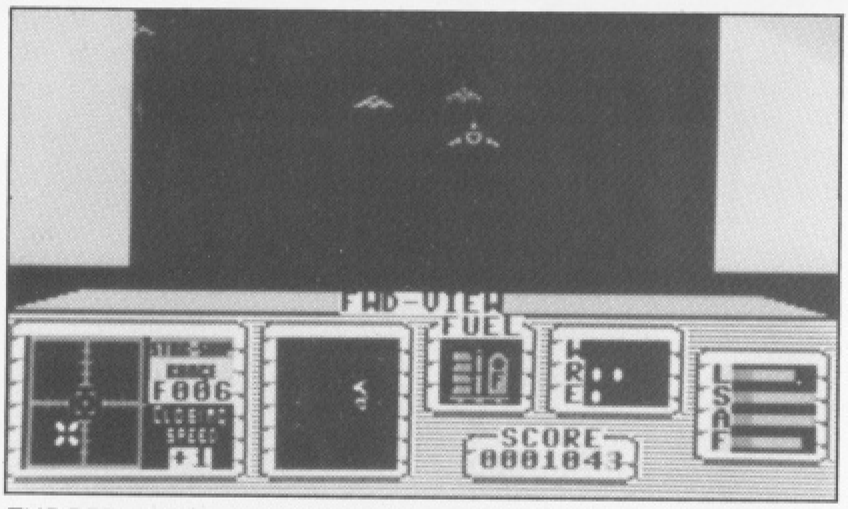
The graphics in Willow Pattern are exquisite with pictures of Chinese temples, bridges and trees. My first impression on playing it was that it was a Sabre Wulf look-alike but the way that the game plays makes it totally different. At £3.95 it is excellent value for money.

**G.R.H.**

## Space Hunter

**Mastertronic £1.99 C64 + Joystick**

Originality 2 | Playability 6 | Graphics 6 | Value For Money 5




THE RED ALERT FLASHES AND you prepare to defend yourself against the waves of fighters that threaten to destroy you.

The inhabitants of Earth are starving and none of the space fleet pilots have returned from their mission in search of food. As a last desperate attempt to save the Earth you, a Rookie pilot, have been sent to capture the food transporters before the aliens.

The position of the target ship, which can be changed by the player, is indicated by a flashing dot on the radar. Use the warp drive or forward thrust to speed you to the target ship, as loitering brings hordes of horrible fighters. Once close enough, an approach speed of one or two will automatically teleport you into the ship. Now with the aid of a jet pack you can start your search for extra 'warp drives', flashing food units, more fuel and shields. Beware of the ricochetting creatures which will deplete your energy and kill you if you collide with them. Once you have collected all you can, exit and find the next target ship.

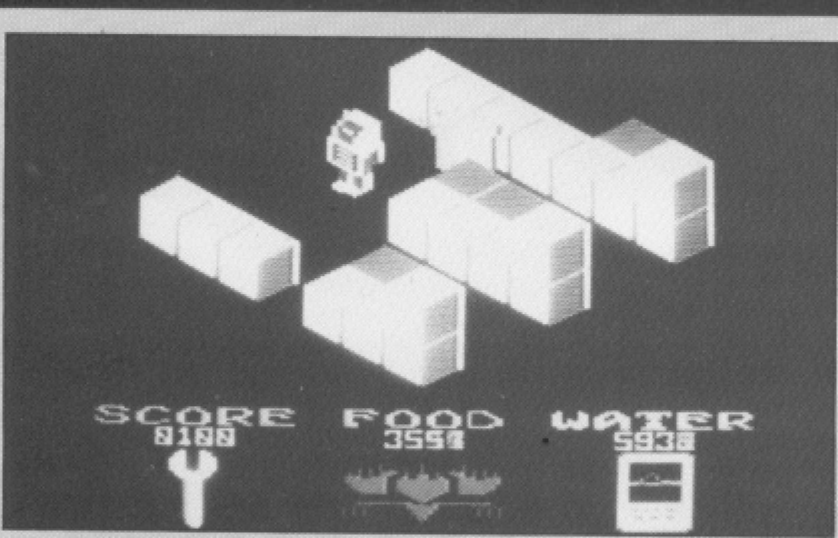
For a cheap game styled on Elite this is not at all bad. To progress through the 11 ranks from Rookie to Space Hunter will not take an experienced Elite-ist too long. The documentation, as in many Mastertronic games is not very comprehensive. The graphics are realistic and the music is pleasantly relaxing after your ordeal with the fighters.

**J.L.**

## Chimera

**Firebird – Super Silver Range £3.95**

Originality 4 | Playability 7 | Graphics 7 | Value For Money 7




THAT THE CRAFT ORBITING the earth is hostile is undeniable. It also appears that someone is going to have to find out how much of a threat it represents and if possible destroy it. If you escape with your life, so much the better!

As the airlock shuts behind you, you look around and see that you are in a large chamber with exits left and right and an exit in front of you blocked by what appears to be a giant microchip. The rooms are depicted in 3D block graphics, reminiscent of Ultimate's Alien 8 and Knightlore for the Spectrum. The effect works well.

Destruction of the ship requires a four stage sequence and your only clue is that the first stage requires a spanner so it seems reasonable to set off looking for one. There is nothing much to stop you as you explore – nothing to shoot – but certain areas are restricted to you at the start and kill you if you try to enter them. Time however is against you and your supplies of food and water soon disappear and have to be replenished as you find them.

You die if either level reaches zero. Objects are frequently hidden behind corners 'that you can't see' and so detailed exploration is essential. A scrolling message keeps you informed of what's going on.

Chimera is an enjoyable arcade adventure but it lacks the sophistication of some of its bigger brothers.

**G.R.H.**

## Friday the 13th

**Domark £8.95 C64 + Joystick**

Originality 7 | Playability 6 | Graphics 6 | Value For Money 7




FRIDAY THE 13TH IS BASED on the scythe-fi movie of the same name. It is a tale of twisted revenge as homicidal maniac Jason tries to avenge his mother's murder.

His wrath is turned against the innocent holiday makers on the shores of Crystal Lake and the field of action covers 30 screens with blood. In addition there are three buildings: a church and a barn (four screens each (plus a bungalow (12 screens). This gives Jason plenty of room to cleave.

The computer selects a character for you to play and it is your duty to look after the other players by herding them all into a room where you have placed a sanctuary cross.

At some point you may have to arm yourself with one of the weapons lying around and tackle Jason in combat. If you succeed in killing him don't feel too secure, just when you think you have seen the last of him, up he pops again feeling pretty cut-up about his last encounter.

The game is quite challenging to play and the graphics are blocky but pleasant, I don't really think that the quality of the game will really have as much sway with sales as the horrific subject matter. The playing instructions give hints on how to get the most from the horrifying screaming sound effects.

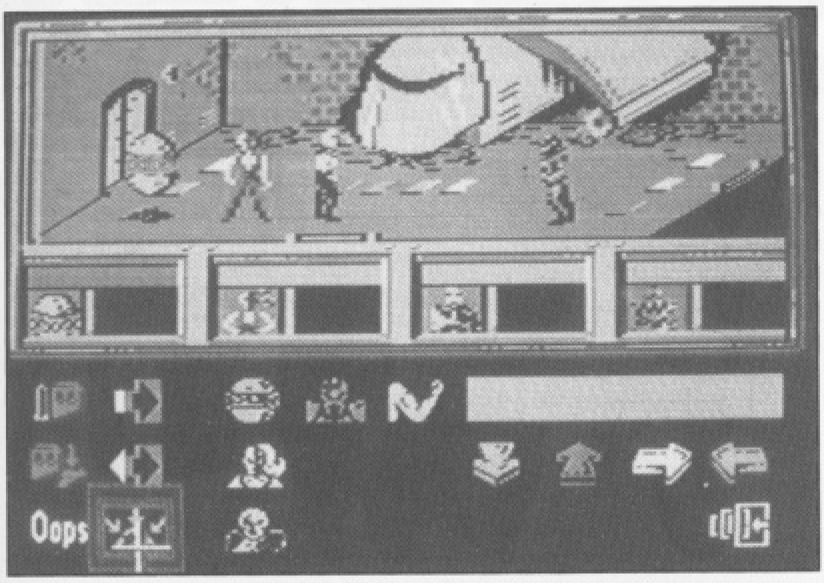
A gory story for the bloodthirsty buyer and just to quench that thirst you get two foaming blood capsules with the game.

**E.D.**

## Enigmaforce

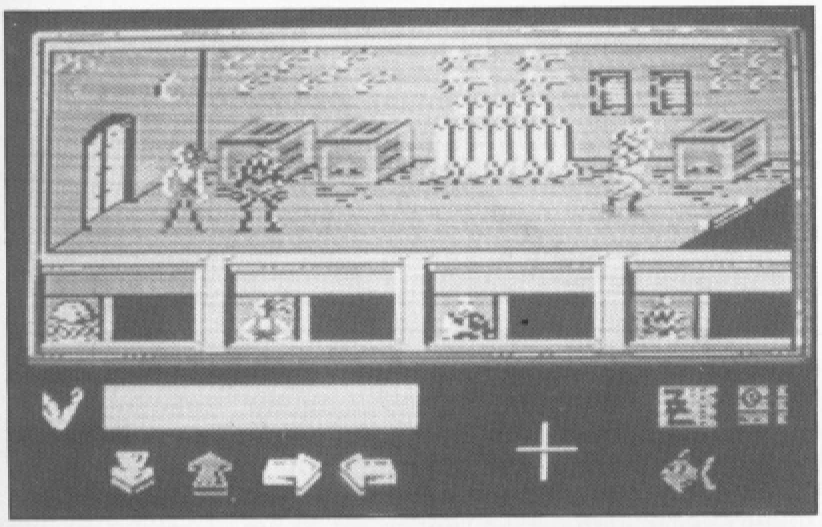
**Beyond Software £9.95 C64 + Joystick**

Originality 9 Playability 9 Graphics 9 Value For Money 9




THE E-TEAM ARE BACK WHEN Enigmaforce try to overcome the deadly General Zoff. This is the expected sequel to Shadowfire and the gameplay is even more advanced than before.

Having captured Zoff in the previous game, four members of the Force were accompanying Zoff to his trial when their ship crashed on team member Syylk's home planet after collison with a gravity mine. The game starts as the Enigmaforce members regain consciousness and discover that Zoff has made off.



Syylk's insectoid people have enough to worry about with an attack from reptiloide so the team has to enlist the help of the insectoid commander. This involves the team in much frantic searching and the game allows you to disperse your force as you see fit. Each member can be located and relocated whenever necessary and the activity selection panel on the lower half of the split screen is used to issue commands.

Enigmaforce consists of Zark Montor the team leader, Syylk the strong warrior, Sevrina Maris the markswoman and locksmith, and last but not least Maul the combat Droid.

As the team goes from location to location they can pick up objects which are lying there but care has to be taken to ensure that each player's special skills are catered for. When the commander has been located, the reptiloid zone must be searched for the only serviceable spacecraft on the planet. Zoff is also searching and must be stopped and captured before Republican destructor tugs arrive to blast the planet to smithereens.

The activity panel is fairly complex and it takes a little time to familiarise yourself with it. At the top is a picture of each team member and if the crosswire selector is placed over one of these panels and the fire button pressed then you are immediately shown that team member's current location. You can now use the arrow icons to dictate a direction for that character to take or select a sequence of actions for them to perform. At first it is relatively simple to use the panel but as the game progresses and the action hots up, you have to keep your wits about you and your joystick on the move.

Action commands are like an icon driven version of an adventure command. For example, to get Maul to pick up explosives you first select Maul's icon panel. Then check the inventory of objects around to see that the explosives icon is there and place the crosswire over the pickup icon. If the button is pressed when the crosswire has been moved over the explosives icon, it will move across into the panel showing the list of objects carried by Maul.

As you can guess the game and the graphics are very sophisticated and put many similar multiscreen games to shame. It just shows what can be done with the 64 in the hands of an intelligent programmer. Smooth animation of each character's movement adds cartoon realism to the game and the excitement of the many skirmishes should hold your attention for hours.

**E.D.**

## The Human Race

**Mastertronic £1.99**

Originality 8 Playability 7 Graphics 6 Value For Money 9




THE HUMAN RACE TOOK 350,000,000 to develop years and believe me it will take you as many hours to complete this game. The idea is quite simple – you have to develop a rather ugly and stupid ape into a man, remember Darwin's theory? This is achieved by completing each screen after which the ape slowly develops its manly features.

The game begins in prehistoric times complete with a large stegasauras, dragonfly, pterodactyl and lots of bananas. Your task is to get the ape to the largest banana whilst avoiding contact with the roaming dinosaurs. The route to the banana appears very straight-forward but it requires some thought if you are to reach it. Once completed chapter two begins.

Here the ape is sent forward to the age of lava and fireballs. His task is to reach the end of a moving pathway without falling off or getting hit by a hurling fireball. This screen belongs in something like 'Journey To The Centre Of The Earth'!

As the game continues the tasks become very difficult but still enjoyable and certainly provide a good challenge.

The graphics aren't the best I've seen and there is a rather annoying delay every time a life is lost but generally the game provides excellent value for money.

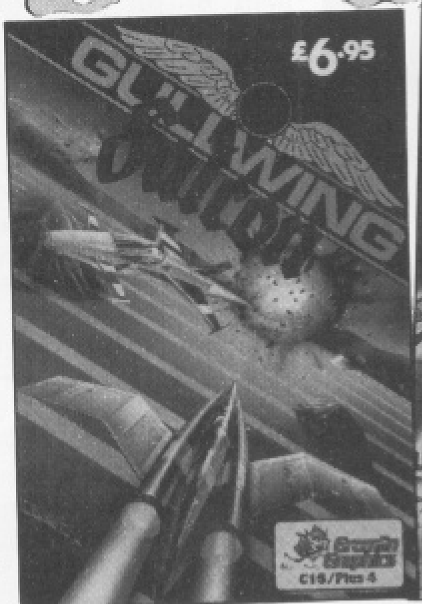
Go ahead and buy it and watch out for a lovely loincloth that would be worthy of any Tarzan!

**S.K.**

**You don't have to be mad to work here, but it helps. Marie Curry visited Ian Stewart at Gremlin Graphics.**



# Gremlin GRILLING










THERE'S A LITTLE GREEN MAN ALIVE AND well and living in Sheffield. This pea-coloured gremlin has made his home there since June 1984 and the climate seems to agree with him.

Gremlin Graphics is a small compact outfit run by the exuberant Ian Stewart who defines himself as the inspiration behind many of his company's products. Entering the micro industry through Laskys' first computer outlet in Sheffield, he soon realised that there were opportunities for a bright young man in this youthful business.

Inspired with this confidence, Ian opened Sheffield's first specialist computer shop aptly named, Just Micro. The venture took off and Just Micro did just fine. It was less than a year later that the gremlin got into the works.

In the summer of 1984, Ian and partner, Kevin Norburn, took the plunge, caught the gremlin and stuck him on some headed notepaper to form Gremlin Graphics. Once this was done there was no going back and Wanted: Monty Mole, Gremlin's first game, was launched amid a blaze of publicity. The miners' strike was then entering the long and not so hot summer and the Gremlin Graphics boys saw the potential of Arthur Scargill's activities as material for a computer game storyline. Because of its topical content the game was treated to wide coverage on both national television and radio and became a chart success paving the way for two subsequent Monty games.

Ian Stewart believes that a major reason for Gremlin Graphics's continued success in the production of popular games is the connection which is closely maintained with Just Micro. All Gremlin's games are extensively tested on unsuspecting members of the public who innocently venture into the shop. "Customers are excellent bug fighters." commented Ian. "Our programmers may test a game for days without finding anything, then one of the kids will come across a bug in a few minutes of play."

The Gremlin outfit listens very carefully to the comments of the customers who try out new products. Ian crisply summed up the Gremlin policy in this area: "If we don't get the reaction we want then we scuttle it!" Simple but effective.

Many Gremlin game ideas are built up around a comical central character and this seems to have become a successful technique. Names such as Potty Pigeon, Sam Stoat Safebreaker and Thing on a Spring come to mind. "We like to base a game around a particular character to stimulate the imagination of the player," said Ian. "We tend to concentrate on arcade games because they give an outlet to the sense of humour which is a ruling principle at Gremlin. Wherever I am, I'm always thinking of new ideas for games."

There are only four full time programmers at Gremlin, other work is done by about 10 regular freelancers. Pete Harrap who wrote the original Monty Mole is now a permanent member of staff. His association with Ian Stewart began when his Spectrum broke down and Just Micro lent him another until it was repaired. Ian said: "Pete was a customer in the shop and we realised he had great programming ability. We lent him the Spectrum and it just went on from there." However, Sheffield natives with defunct computers shouldn't rush round to try and get a loan of a machine.

The full time programmers at Gremlin are now collaborating to produce a series of games based on the popular role-playing books: The Way of the Tiger. The books revolve around the adventures of Avenger, a Ninja warrior. Work on the series is progressing fast and the first part should be available in February. Although programmers at Gremlin have always worked together to a certain extent this is the first project into which the team have ploughed their joint efforts from its inception and the co-operation between them has been extremely successful. Praising his boys, Ian said: "Areas which need improvement can be sorted out through collaboration. There's never a cross word between them." There will be four games in the series when it is completed and they will all follow a similar story line to the books.

Another new game, Bounder, features the incredible antics of a bouncing tennis ball. It's certainly a different concept for a game as the ball leaps toward you out of the screen and then bounces back into the distance. Gremlin's programmers have even put spin on it! According to Ian the game comes in a value for money pack with another game, entitled Metabolis, on the back of the cassette.

C-16 and Plus/4 games are an important part of the Gremlin range and owners will be pleased to know that Gremlin has no intention of discontinuing this commitment to these machines. There are probably around 80,000 C-16 and Plus/4 owners in the country now and there are very few software houses which take notice of them. Looking at the situation in the cold light of day Ian remarked: "The less people that produce games for the C-16 and Plus/4, the better for Gremlin." A recently released compilation contains four games for £9.95 so the much neglected users can really get zapping.

Ian obviously believes he's got his policy right but he stresses the fact that above all the computer industry is a lot of fun. "The main strength of Gremlin is the atmosphere in the firm. This makes for a good working situation and consequently good products." he remarked almost drowned out by the zapping noises from next door and the blare of electronic music from a neighbouring office.

When asked to sum up Gremlin as a software house, Ian makes the place sound more like a lunatic asylum. "Basically we're a barmy lot up here."

# Missives

**Yet another bundle of replies to your letters.**

## Problems, Problems

I WAS DELIGHTED WHEN THE PROGRAM *System 64* by J A Wolfe appeared in the August issue of the magazine. I was able to enter all the addresses of my associates and it worked well. That is to say until this week!

I wanted to amend the address of a colleague who had moved house. So, using option 2 'Delete Record', I cancelled the old address and re-entered the new address on the end of the list of addresses.

Imagine my disappointment when I printed out a fresh list of addresses to find that every single address after the one I had deleted now had the wrong name. It appears that the deletion in your program only erases the name and not the whole address. I have studied the program listing but have been unable to amend it. Can you help?

**D W Passmore, Sidmouth**

I have typed in the Home Accounts program from your magazine. However I cannot seem to get it to work.

I have checked my program against the one printed in the magazine and can't find any errors. I haven't seen any corrections printed for this program but I believe that the error is yours and not mine.

Why don't you check your programs before you print them in the magazine as it would no doubt solve many problems?

**A Sherwood, Bridlington**

*Every post delivered to the Your Commodore office is guaranteed to have quite a large number of letters similar to the ones above. So it is probably worth making a few points clear.*

*Every program that we publish in the magazine is very thoroughly tested before we print it.*

*The listings in the magazine are not typeset in any way. What we actually do is get a printout of the program which is then placed on the page as artwork. This leaves very little chance for errors to occur.*

*Obviously errors do creep in sometimes. When they do we always publish corrections in the magazine. Corrections don't usually appear in the next issue of Your Commodore but in one after that. This is because when one issue is on sale another is just about ready to be printed.*

*Most of the time the errors are made by the person typing the listing into the machine. No matter how many times you check them little errors do slip through. In order to help you with this we will offer to send a new computer listing to anyone having problems with a program if they send us a stamped self-addressed envelope and state clearly what they require.*

*In the near future we are hoping to start a software service where all of the programs in a particular issue will be available on cassette but we have no firm date for this as yet.*

*In the meantime a couple of features that will try to give you some hints about de-bugging programs have been commissioned. And, being honest, fault finding a program is an extremely good way of learning about programming. Perhaps we should start a spot the deliberate mistake page!*

*Going back to the System 64 program. It appears that the programmer made a few mistakes when he wrote the program. He only deletes the name and not the address. This is done in the Sort routine. This is the type of mistake that it is not always possible to spot so please make sure that you give any programs sent to the magazine a thorough testing before you stick them in the post. Anyway here are the lines that will need to be added to delete a whole record.*

```
3562 LET L2$(L)=L2$(T):LET L2$(T)="*"
3563 LET L3$(L)=L3$(T):LET L3$(T)="*"
3564 LET L4$(L)=L4$(T):LET L4$(T)="*"
3565 LET L5$(L)=L5$(T):LET L5$(T)="*"
3566 LET L6$(L)=L6$(T):LET L6$(T)="*"
```

## Too Many Ads?

I am writing to complain about the recent change which has occurred since the merging of Your Commodore and Your 64. It has come to my attention that there has been a substantial increase in the amount of advertisements, at least 25% in the December issue. As I subscribe to your magazine I am worried that the amount of advertisements may increase even further in the future.

I would also like to see in the magazine an extra page of Flippo (In Arcadia) in exchange for one less page of Sense of Adventure, as I feel that Flippo is far more interesting.

A suggestion I have for your monthly competition is, perhaps, that you should ask people to send in the best one line program or the best game or utility using no more than 100 linnes since a spot the difference competition does not involve the use of a computer in any way.

I hope that the points I have brought to your attention will help to make your magazine even more absorbing.

**Raymond Webb, Guestling**

*Thanks for your comments, Raymond. It's always interesting to find out what people think of the magazine. Your main worry seems to be concerning advertising. This varies greatly throughout the year and tends to rise at Christmas because manufacturers want to make the most of the extra money we all spend. Advertising forms a large part of our income and it is therefore essential that we carry a certain amount in order to keep up the standard of the magazine as a whole.*

*As an arcade game fan, we can bring you some good news. In the near future we will be taking some more ideas from Your 64 to improve our arcade coverage. However, adventure players needn't worry because, you won't suffer from this at all.*

*As for your comment on our competitions, Raymond. We do try and run the occasional different one, but the beauty of a spot the difference competition is that anyone can enter, and they needn't have any knowledge of programming to do so. Since the prizes are usually games of some sort, then it seems illogical to set a problem which only programmers can answer.*

# LANGUAGE LAB — PILOT

**David Janda takes control and guides you through Commodore's Pilot package.**

THIS MONTH'S LANGUAGE LAB IS rather different from those in previous issues. Presented here is a brief introduction to a language that is very popular in the educational field in America. No, it's not Logo, but Pilot, and in my opinion, it would be just as popular if it was given the amount of attention it deserves. Commodore Pilot is the only package which is being reviewed, and the surprising thing is that the Commodore version of standard Pilot (known as common Pilot) includes many facilities not found in the standard.

It is worth pointing out that although the language is designed to be used by educators to write interactive educational programs, it does have its attractions for the programmer. The reason I say this is because of pattern matching.

It can be said that the job of a programmer is to solve the problem at hand. In educational programs this can be quite difficult as there is a lot of user input which has to be interrogated. This is not so easy, and programmers normally find themselves spending a lot of time writing routines that interrogate user input. This can distract them from the job at hand – i.e. solving the problem!

Pilot incorporates a number of tools that facilitate the interrogation of user input, thus leaving the programmer to get on with the main task. These tools can be applied to problems associated with string manipulation and pattern matching.

## Pilot — The Language

Pilot is a computer assisted instructional (CAI) language, which is designed for teachers to produce educational programs – courseware as the manual terms it. The idea behind Pilot is that it should be easy for teachers to produce programs that interact with the student on a question and answer level.

The version which is supplied on disk will run Common Pilot programs, and do a lot more since extensions have been added. These allow for the use of colour, graphics and sprites. An example is that text windows may be set up to allow for questions and answers to be displayed on different sections of the screen.

Graphics are catered for with commands to allow lines to be drawn/plotted as well as filled in with colour. A group of sprite and colour commands is also included, and the extensions facilitate the use of the advanced features of the C64.

The syntax of Pilot is very simple. Pilot instructions consist of several parts. First, there is a one-letter opcode (of which there are 20. The opcode is then optionally followed by a modifier which changes the way the opcode is going to be executed. Conditions can then follow and they can determine whether the instruction is to be carried out. This is best described with an example:

```
TS(A=5): Correct, the answer is 5.
```

Here the opcode T means print something, but the screen is first cleared with the modifier S. The condition is that the answer A equals five, and if so the text in the field (everything following the colon) will be printed.

Unlike Basic, Pilot does not require strings to be enclosed in quotes. Instead, the string variable or literal is placed after the separator. This very simple Pilot program demonstrates this:

```
T: This will be printed
```

This will simply display "This will be printed" at the current cursor location. However, a return is also printed after each occurrence of the T command. It is possible to 'hold', or keep the cursor on the same line by using the H modifier.

```
TH: What is your name
```

This will display the message, and any answer will be entered on the same line (it is important to leave two spaces after the message).

## Pilot Data Types

Maths in Pilot is integer only which is a bit of a restriction. The range is between -32768 and +32767. Another restriction is that only 26 variables are allowed for.

Performing calculations is done with the computer instruction which takes the form of C. Variables assignment is performed after the separator and printing the values of variables requires a £ character to precede the variable name.

```
C:A=2+2 — assigns A with 4
C:B=(2+2)*(3+1) — assigns B with 16
T:Here is a sum...
TH:2+2=
C:A=2+2
T:£A
T:There, the answer is £A
```

Strings are handled in a very flexible way. But, as mentioned Pilot only has 26 variables available so it is not possible to have a string and a numeric variable of the same letter.

Before a variable can be used as a string, it must first be dimensioned with its maximum length. Pilot allows a maximum string length of 255, and the command used to dimension string is D. The $ character is used to identify that the variable is a string, but this is not absolutely necessary.

```
D:A$(10)
```

The computer command is used to assign a string variable with a value. In this case, it is necessary to use quotes. When using the T command to print the contents of a string variable a $ character is used to precede the variable.

```
D:A$(14)
C:A$="Your Commodore"
T:This magazine is called $A$
```

The C command is very flexible when it comes to string assignment. Strings can be assigned with sub-strings, concatinated and so on. Indexing is allowed which can be used on the source or object string.

```
D:A$(18)
D:B$(20)
C:A$="This is    funny"
C:B$="He is not very funny"
C:A$(9,3)=B$(11,4)
T:$A$
```

Would print: "This is very funny".

Concatination is also possible using the !! operator.

```
D:A$(6)
D:B$(7)
D:C$(13)
C:A$="Hello"
C:B$="readers"
C:C$=A$!!B$
T:$C$
```

Would print: "Hello readers".

Getting user input into the micro is very simple in Pilot. A pre-defined input buffer called %B is used to store user input. It works like this:

```
T:Hello, who are you?
A:
T:Pleased to meet you $%B
```

Notice that like ordinary string variables, the input buffer requires the $ to be prefixed to the buffer name when it is being printed.

The A is the accept command, and it can work with numeric and string variables as well as the input buffer variable:

```
D:A$(10)
TS:What is your name?
A:$A$
T:And how old are you $A$ ?
A:£B
T:You are £B years old then $A$
```

The problem with user input is that you don't always get it in the format you wanted. Some people would enter their names in upper case, some lower, and some as a mixture of both. This can be a real headache especially if the input is to be processed. Pilot provides a problem command PR that allows input to be 'converted' into a specified choice. PR:U will convert all input to upper case, PR:L to lower and PR:S will strip any input of spaces. PR:Z will reset the options.

## The Clever Stuff

Pattern matching is achieved with the match command M. Assuming we wanted to check that the user reads this mag, we could pose the question and process the answer using this program:

```
T:What Commodore magazine do you
read?
M:Your Commodore

A:
```

Now, if the answer entered was "I read a magazine called Your Commodore which I think is great", believe it or not a match would be made. This is because Pilot does the hard work involved in searching (called window searching) though the users input checking to see if there is a match.

Problems with this are that the user may enter the answer in upper or lower case. To still get a match the PR command would be used before the match command to convert the input.

Match used with the S modifier will even accept answers that are spelt incorrectly! The S modifier will accept an answer even if one letter is wrong, or if a pair of characters have been reversed – very handy!

More flexibility is allowed with the * and & characters when used in the match command:

```
M:Comm*dore
```

This simply means 'accept any letter in place of the *'. The & means any number of characters.

## Summary

There are many, many other features to the Pilot language. Jumping, subroutines, multiple choice tests, even hinting is possible. All these features make Pilot a very practical tool for educational programming.

The language does suffer in some areas though. The restriction on the number and length of variables is a serious one, as is the lack of floating point maths. But these faults are common to the Pilot standard, and are not unique to Commodore's implementation.

Even though the language is very powerful, it is not very hard to learn. Commodore Pilot has 20 commands with modifiers and because the syntax is very simple, it is quite possioble to write complex programs in a very short time.

I would strongly recommend Pilot to anyone who wishes to write programs that involve processing interactive answers. The features available in the language enable the programmer to 'get on with the job'.

## Commodore Pilot

Commodore Pilot is supplied on disk only, together with a very good 111 page tutorial/instruction manual. Two versions of the Pilot interpreter are supplied on the distribution disk; a development version which is used to write, edit and run Pilot programs, and a run only version that is identical except programs can only be loaded and run.

Other files on the disk include three demonstration programs, a simple sprite editor (written in Pilot), and a Pilot program that enables the user to experiment with sounds on the C64

The actual Pilot package has four modes of operation. First there is the edit mode which is used for program creation and editing. The run mode is for running the program, and the command mode is used for loading, saving and printing programs. Finally, the immediate mode (which is similar to Basic's immediate mode) allows the programmer to experiment with Pilot by trying out Pilot commands one at a time. This mode is very handy when learning the language.

Graphics on a 320×200 grid are catered for. Points can be plotted and removed, lines drawn and the graphics origin changed. Both text and graphics can be freely mixed and a split-screen command allows the screen to be divided between graphics/text output and prompts/user input.

Other features include sound, sprites and user definable characters, although I must say that these could have been implemented in a more friendly way.

Commodore Pilot not only conforms to the standard common Pilot, but also includes many new features (some of which I have mentioned). The package is an easy one to use which is a blessing. Highly recommended!

# PROGRAMMER OF THE YEAR

commodore

**This month's entry is Spike, an excellent game by Shane Stevens.**

A LARGE NUMBER OF GAMES have been entered for the Programmer of the Year Competition. Spike is certainly an above average entry. It is definitely worth the effort of typing it in.

In the game you find yourself as Spike travelling around a Power Grid. Hidden somewhere within the grid is your trusty C64. Your job is to find it.

Of course, life isn't easy and the Sparks brothers are out to get you. The number of sparks charging around the grid depends on the level at which you choose to play, there are nine in all.

Full playing instructions are included in the game so there is no point giving them here.

## Getting It All In

Spike is in two parts. The first part is in Basic and should be typed in and saved on to tape or disk. Make sure you read the page that tells you all about our method of printing listings before you start (I don't want [HOME] – Ed).

If you are using a cassette then make sure that you change the

LOAD "SPIKE",8,1

in line 40 to:

LOAD "SPIKE",1,1

Once you have SAVEd the Basic you can then tackle the machine code. Yes, I know that there's a lot of it but we have tried to make it as easy as possible for you.

Elsewhere in this magazine you will find the Your Commodore Easy Entry program. You should type this in and save on something safe. You will need this for most of the machine code programs in Your Commodore. RUN this and follow the instructions with the Easy Entry article. Don't forget each line is checked as you type it in and you can SAVE what you have entered at any time.

Make sure you SAVE it before you attempt to RUN it.

Spike should be SAVED straight after the SPIKE LOADER and is SAVEd between the following locations:

Start Address : 32768
End Address : 37296

Remember to press F1 in the Easy Entry program to activate the SAVE routine, and make sure you save the program with the name SPIKE.

## And On We Go

Now that you have both parts stored on tape you simply have to LOAD and RUN the SPIKE LOADER program. This will automatically LOAD the second part and the game will start to RUN.

Have fun!

### Program: Spike Load

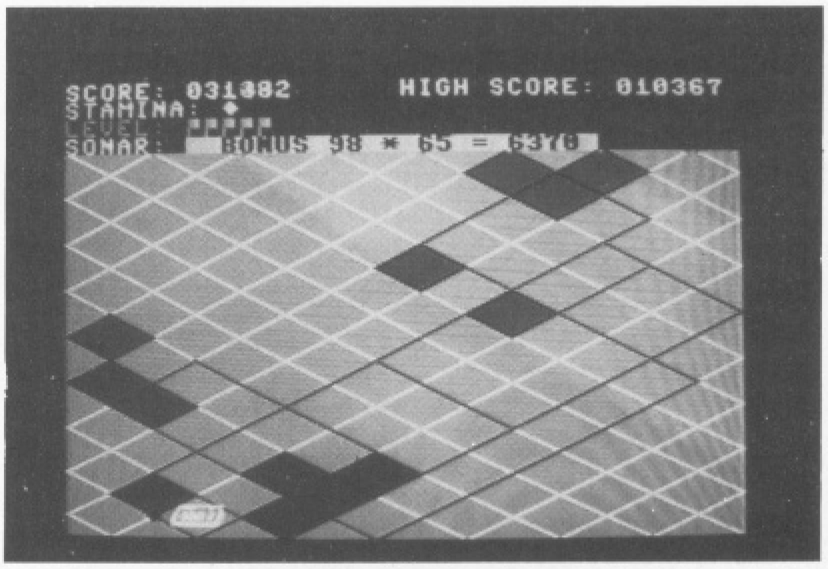
```
1 REM SPIKE BY SHANE STEVENS
   85'
2 IF A=1 THEN GOTO 20
4 A=A+1:LOAD "SPIKE",8,1
   :REM CHANGE TO ,1,1 FOR
   CASSETTE
20 S=54272:FOR L=S TO S+24
   :READ DA:POKE L,DA:NEXT
30 DATA 96,22,0,1,0,35,166,
   195,16,0,1,0,35,166,48,4,
   0,8,0,35,166,0,15,244,31
40 FOR AD=49160 TO 49202
   :READ DA:POKE AD,DA:NEXT
50 DATA 120,173,224,3,174,
   225,3,172,226,3,141,4,212,
   142,11,212,140,18,212,88,
   96
60 DATA 206,224,3,206,225,3,
   206,226,3,32,8,I92,238,
   224,3,238,225,3,238,226,3,
   96
70 POKE 992,67:POKE 993,65
   :POKE 994,33
90 POKE S+18,33:FOR T=1 TO 1
   30:NEXT:POKE S+18,32
   :POKE S+14,180
   :POKE S+18,33
95 FOR T=1 TO 130:NEXT
100 POKE S+18,32:POKE S+22,1
   :POKE S+14,251
   :POKE S+18,33:FOR T=15 TO
   150:POKE S+22,T:NEXT
110 POKE S+18,32:POKE S+22,90
   :POKE S+14,71:POKE S+15,5
   :POKE S+18,33:FOR T=1 TO
   130:NEXT
120 POKE S+18,32:POKE S+14,
   152:POKE S+18,33
   :FOR T=1 TO 140:NEXT
   :POKE S+18,32
130 POKE S+14,237:SYS 49160
   :FOR T=1 TO 180:NEXT
   :POKE S+3,8:POKE S+10,8
140 FOR T=1 TO 180:NEXT
   :SYS 49181:FOR T=1 TO 170
   :NEXT:POKE S+3,1
   :POKE S+10,1
150 SYS 49160:FOR T=1 TO 80
   :NEXT:SYS 49181
   :FOR T=1 TO 170:NEXT
   :POKE S+3,8:POKE S+10,8
160 SYS 49160:FOR T=1 TO 80
   :NEXT:SYS 49181
   :FOR T=1 TO 170:NEXT
   :POKE S+3,1:POKE S+10,1
170 SYS 49160:FOR T=0 TO 170
   :NEXT:POKE S+3,8
   :POKE S+10,8:POKE S+14,71
   :POKE S+15.6

180 FOR T=1 TO 100:NEXT
   :SYS 49181:FOR T=1 TO 270
   :NEXT:POKE 992,17
   :POKE 993,17
185 A=50:B=37:C=31
   :POKE S+6,249:POKE S+13,
   249:POKE S+20,249
   :POKE 994,17
190 POKE S,60:POKE S+1,A
   :POKE S+7,162:POKE S+8,B
   :POKE S+14,165:POKE S+15,C
200 SYS 49160:FOR T=1 TO 200
   :NEXT:SYS 49181
   :FOR T=1 TO 21:A=A-1:B=B-1
   :C=C-1:POKE S+1,A
210 POKE S+8,B:POKE S+15,C
   :POKE S+14,48:POKE S+15,4
   :POKE S+20,242
215 POKE S,119:POKE S+1,7
   :POKE S+7,251:POKE S+8,4
   :POKE S+6,165:POKE S+13,
   165
217 GOSUB 220:GOTO 280
220 POKE S+18,65:FOR T=1 TO
   170:NEXT:POKE S+17,1
   :FOR T=1 TO 75:NEXT
   :POKE S+18,64
230 FOR T=1 TO 170:NEXT
   :POKE S+17,8:POKE S+18,65
   :FOR T=1 TO 75:NEXT
   :POKE S+18,64
240 FOR T=1 TO 170:NEXT
   :POKE S+17,1:POKE S+18,65
   :FOR T=1 TO 75:NEXT
   :POKE S+18,64
250 FOR T=1 TO 170:NEXT
   :POKE S+17,8:POKE S+18,65
   :FOR T=1 TO 75:NEXT
   :POKE S+18,64
260 FOR T=1 TO 170:NEXT
   :POKE S+17,1:POKE S+18,65
   :FOR T=1 TO 75:NEXT
   :POKE S+18,64
270 FOR T=1 TO 170:NEXT
   :POKE S+17,8:POKE S+18,65
   :FOR T=1 TO 75:NEXT
```

```
 :POKE S+18,64:RETURN
280 POKE S+4,33:POKE S+11,33
 :FOR T=1 TO 300:NEXT
 :POKE S+4,32:POKE S+11,32
290 POKE S,97:POKE S+1,8
 :POKE S+7,152:POKE S+8,5
 :X=25:Y=18:Z=15
300 POKE S+4,33:POKE S+11,33
 :FOR T=1 TO 100:NEXT
 :POKE S+4,32:POKE S+11,32
310 FOR T=1 TO 180:NEXT
 :POKE S+14,35:POKE S+15,3
 :GOSUB 220:POKE S+18,32
 :FOR T=1 TO 30:NEXT
320 POKE S+18,33:FOR T=1 TO
 100:NEXT:FOR X=3 TO 255 S
 TEP 2.5:POKE S+15,X:NEXT
350 POKE S+14,48:POKE S+15,4
 :GOSUB 220:POKE S,251
 :POKE S+1,4:POKE S+7,247
 :POKE S+8,9
360 POKE S+4,33:POKE S+11,33
 :FOR T=1 TO 300:NEXT
 :POKE S+4,32:POKE S+11,32
370 POKE S,97:POKE S+1,8
 :POKE S+7,152:POKE S+8,5
 :X=25:Y=18:Z=30
380 POKE S+4,33:POKE S+11,33
 :FOR T=1 TO 100:NEXT
 :POKE S+4,32:POKE S+11,32
390 FOR T=1 TO 180:NEXT
 :POKE S+14,35:POKE S+15,3
 :GOSUB 220
400 POKE S+6,249:POKE S+13,
 249:POKE S+20,249
410 POKE S,30:POKE S+1,X
 :POKE S+7,209:POKE S+8,Y
 :POKE S+14,210:POKE S+15,7
420 POKE S+3,8:POKE S+10,8
 :SYS 49160:FOR T=1 TO 200
 :NEXT:SYS 49181
 :FOR T=1 TO 15
430 X=X-1:Y=Y-1:Z=Z-1
 :POKE S+1,X:POKE S+8,Y
 :POKE S+15,Z:NEXT
 :POKE S+20,241
440 POKE S+14,48:POKE S+115,4
 :GOSUB 220:POKE S,251
 :POKE S+1,4:POKE S+7,125
 :POKE S+8,2
450 POKE S+4,3:POKE S+11,33
 :FOR T=1 TO 250:NEXT
 :POKE S+6,242:POKE S+13,
 242
455 POKE S+4,32:POKE S+11,32
 :FOR T=1 TO 10:NEXT
460 POKE S,97:POKE S+1,8
 :POKE S+7,152:POKE S+5,5
470 POKE S+4,33:POKE S+11,33
 :FOR T=1 TO 100:NEXT
 :POKE S+4,32:POKE S+11,32
480 FOR T=1 TO 180:NEXT
 :POKE S+14,35:POKE S+15,3
 :GOSUB 220:POKE S+18,32
 :POKE S+23,244
490 POKE S+6,247:POKE S+13,
 247:POKE S+20,247:A=200
 :B=150:C=126:FOR T=1 TO 1
 30:NEXT
500 POKE 992,17:POKE 993,17
 :POKE 994,17:POKE S+18,65
 :FOR T=20 TO 120 STEP 2
 :POKE S+22,T
505 NEXT:FOR T=120 TO 20 STE
 P-2:POKE S+22,T:NEXT
 :POKE S+18,64:FOR T=1 TO
 100:NEXT
510 POKE S,165:POKE S+1,A
 :POKE S+7,96:POKE S+8,B
 :POKE S+14,225:POKE S+15,C
511 POKE S+23,4:POKE S+22,220
520 SYS 49160:FOR T=1 TO 100
 :NEXT:FOR T=1 TO 186:A=A-1
 :B=B-.75:C=C-.63
 :POKE S+1,A
530 POKE S+8,B:POKE S+15,C
 :NEXT:POKE S+8,22
 :POKE S+15,8:FOR T=1 TO 1
 00:NEXT
540 SYS 49181:FOR T=1 TO 30
 :NEXT:SYS 49160
 :FOR T=1 TO 100:NEXT
550 SYS 49181:FOR T=1 TO 180
 :NEXT:SYS 49160
 :FOR T=1 TO 75:NEXT
560 SYS 49181:FOR T=1 TO 180
 :NEXT:SYS 49160
 :FOR T=1 TO 75:NEXT
570 SYS 49181:FOR T=1 TO 180
 :NEXT:SYS 49160
 :FOR T=1 TO 75:NEXT
 :SYS 49181
580 FOR T=1 TO 40:NEXT
 :POKE S,135:POKE S+1,33
 :POKE S+7,31:POKE S+8,21
 :POKE S+14,97
590 SYS 49160:FOR T=1 TO 70
 :NEXT:SYS 49181
 :FOR T=1 TO 550:NEXT
 :POKE S,15:POKE S+1,67
600 POKE S+7,135:POKE S+8,33
 :POKE S+14,48:POKE S+15,4
 :POKE 992,33:POKE 993,33
610 POKE 994,33:SYS 49160
 :FOR T=1 TO 100:NEXT
 :SYS 49181
700 POKE 53272,(PEEK(53272)A
 ND 240)OR 4
710 POKE 53270,PEEK(53270)AN
 D 239
720 POKE 53265,155
730 POKE 53281,BC:POKE 646,FC
800 POKE 53280,11
 :POKE 53281,12
801 GOTO 840
805 PRINT"[CLEAR,WHITE,UP2,
 SPC16]*******"
810 PRINT"----------------
 [WHITE,CM,RVSON,C8]
 !SPIKE![RVSOFF,CG,WHITE]
 ----------------";
820 PRINT"[WHITE,SPC16,CT7]";
830 RETURN
840 GOSUB 805:PRINT"[HOME,
 DOWN5,SPC6]HELLO AND WELC
 OME TO[SPC,RVSON]!SPIKE!
 [RVSOFF]."
850 PRINT"[DOWN,SPC6]YOU ARE
 A PUNK,HENCE THE NAME"
860 PRINT"[SPC6,RVSON]
 !SPIKE![RVSOFF].YOU MUST
 TRAVEL THE"
870 PRINT"[SPC6]POWER GRID,
 AND AVOID THE"
880 PRINT"[SPC6,RVSON]
 !SPARKS BROS![RVSOFF,SPC]
 WHO GIVE YOU"
890 PRINT"[SPC6]A SMALL SHOC
 K.YOU CAN WTHSTAND"
900 PRINT"[SPC6]FOUR OF THES
 E SHOCKS BEFORE YOU"
910 PRINT"[SPC6]BECOME A[SPC,
 RVSON]FRIZZEL![RVSOFF]
 THE CONTROLS ARE";
920 PRINT"[SPC6]FAIRLY HARD
 TO GET USED TO AS"
930 PRINT"[SPC6]THE SCREEN
 IS TILTED ON A"
940 PRINT"[SPC6]45 DEGREES
 ANGLE AND THEREFORE"
950 PRINT"[SPC6]THE JOYSTICK
 CONTROLS BECOME:-"
960 PRINT"[SPC6]JOYSTICK FOR
 WARD=NE"
970 PRINT"[SPC6]JOYSTICK BAC
 K[SPC3]=SW"
980 PRINT"[SPC6]JOYSTICK LEF
 T[SPC3]=NW"
990 PRINT"[SPC6]JOYSTICK RIG
 HT[SPC2]=SE"
1000 GOSUB 5000:GOSUB 805
1010 PRINT"[HOME,DOWN5,SPC6]
 THIS ADDS A DIFFICULTY"
1020 PRINT"[SPC6]FACTOR TO
 THE GAME.TO COMPLETE"
1030 PRINT"[SPC6]EACH LEVEL
 YOU MUST FIND AND"
1040 PRINT"[SPC6]ENCLOSE THE
 HIDDEN COMMODORE"
1050 PRINT"[SPC6]64 COMPUTER
 IN IT'S GRIDDED"
1060 PRINT"[SPC6]DIAMOND.YOU
 CAN TELL HOW FAR"
1070 PRINT"[SPC6]FROM THE
 COMPUTER YOU ARE BY"
1080 PRINT"[SPC6]THE SONAR
 BAR AT THE TOP OF THE"
1090 PRINT"[SPC6]SCREEN.WHEN
 THE BAR IS SMALL"
2000 PRINT"[SPC6]
 YOU ARE CLOSE AND WHEN
 IT'S"
2010 PRINT"[SPC6]LARGE YOU
 ARE FAR AWAY.IF YOU"
2020 PRINT"[SPC6]ENCLOSE THE
 WRONG GRID IT WILL"
2030 PRINT"[SPC6]TURN BLUE,
 THIS LOWERS YOUR"
2040 PRINT"[SPC6]
 BONUS POINTS AT THE END
 OF THE"
2050 PRINT"[SPC6]LEVEL.YOU
 SCORE 10 POINTS FOR"
2060 PRINT"[SPC6]CROSSING
 A SIDE OF A GRID AND"
2070 PRINT"[SPC6]A BONUS FOR
 FINDING THE COMPUTER."
3000 GOSUB 5000:GOSUB 805
3010 PRINT"[HOME,DOWN5,SPC6]
 AT THE BEGINING OF THE
 GAME"
3020 PRINT"[SPC6]
 YOU CAN SELECT A STARTING
 LEVEL"
3030 PRINT"[SPC6](1-9) AND
 A DIFFICULTY LEVL (E/H).
 ";
3040 PRINT"[SPC6]THESE ARE
 SET TO 5 AND H WHEN"
3050 PRINT"[SPC6]YOU FIRST
 START BUT CAN BE"
3060 PRINT"[SPC6]CHANGED BY
 PUSHING (1-9) OR"
3070 PRINT"[SPC6](E/H) FOR
 EASY AND HARD.ANOTHER"
3080 PRINT"[SPC6]
 USEFUL FEATURE OF[SPC,
 RVSON]!SPIKE![RVSOFF,SPC]
 IS"
3090 PRINT"[SPC6]THE ABILITY
 TO PAUSE THE GAME"
4000 PRINT"[SPC6]BY PRESSING
 THE SHIFT KEY OR THE"
4010 PRINT"[SPC6]SHIFT/LOCK
 KEY."
4020 GOTO 5030
5000 PRINT"[DOWN,SPC6]PRESS
 SPACE BAR TO CONTINUE.
 [UP]"
```

```
5010 GET A$:IF A$<>" "THEN 5
   010
5020 RETURN
5030 PRINT"[DOWN,SPC9]PRESS
    SPACE BAR TO PLAY.[UP]"
5040 GET A$:IF A$<>" "THEN 5
   040
5050 SYS 32768
```

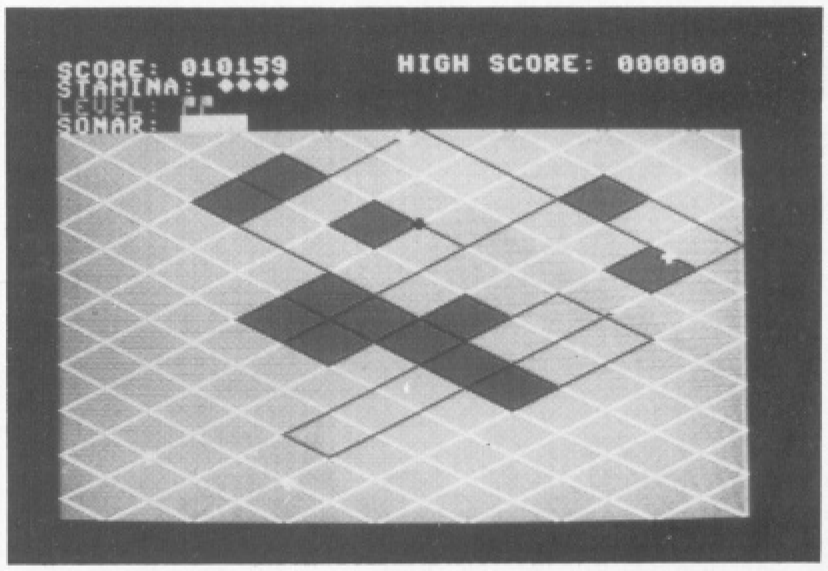
```
32768:169 005 141 190 207 169 072 141 180 207 032 019 252
32780:144 169 007 141 201 207 169 040 141 200 207 169 015
32792:012 141 199 207 169 000 141 039 208 162 024 157 203
32804:000 212 202 224 255 208 248 169 070 141 254 207 178
32816:169 120 141 253 207 169 255 141 015 212 141 182 005
32828:207 169 128 141 018 212 169 064 141 136 002 169 080
32840:001 141 246 207 169 019 032 210 255 169 000 141 126
32852:032 208 173 014 220 041 254 141 014 220 165 001 031
32864:041 251 133 001 160 000 185 000 208 153 000 080 028
32876:185 000 209 153 000 081 185 000 210 153 000 082 086
32888:185 000 211 153 000 083 185 000 212 153 000 084 106
32900:185 000 213 153 000 085 185 000 214 153 000 086 126
32912:185 000 215 153 000 087 200 208 205 165 001 009 036
32924:004 133 001 173 014 220 009 001 141 014 220 169 231
32936:198 141 000 221 169 008 141 024 208 032 183 128 085
32948:076 219 128 120 169 127 141 013 220 169 001 141 168
32960:026 208 169 000 141 018 208 173 017 208 041 127 248
32972:141 017 208 169 119 141 020 003 169 140 141 021 213
32984:003 088 096 032 225 128 076 249 128 169 089 133 096
32996:252 160 000 133 251 169 000 145 251 200 208 251 200
33008:230 252 166 252 224 128 208 243 096 169 016 160 080
33020:000 153 000 064 153 000 065 153 000 066 153 000 035
33032:067 200 208 241 169 022 141 248 067 169 006 153 163
33044:000 216 153 000 217 153 000 218 153 000 219 200 013
33056:208 241 032 182 137 032 043 129 076 187 129 032 180
33068:133 139 169 001 133 002 169 050 141 255 207 172 079
33080:255 207 162 000 032 239 139 232 224 151 240 005 150
33092:136 192 030 208 243 173 255 207 024 105 020 141 010
33104:255 207 201 200 144 225 169 010 141 255 207 174 220
33116:255 207 160 200 032 239 139 136 232 224 151 208 227
33128:247 173 255 207 024 105 020 141 255 207 201 151 042
33140:144 229 169 190 141 255 207 172 255 207 162 000 199
33152:032 239 139 232 224 151 240 005 200 192 200 208 142
33164:243 173 255 207 056 233 020 141 255 207 201 022 105
33176:176 225 169 020 141 255 207 174 255 207 160 030 123
33188:032 239 139 200 232 224 151 208 247 173 255 207 167
33200:024 105 020 141 255 207 201 151 144 229 096 169 126
33212:096 133 252 169 032 133 254 160 000 133 251 133 142
33224:253 177 251 145 253 200 208 249 230 252 230 254 086
33236:166 252 224 127 208 239 177 251 145 253 200 192 086
33248:064 208 247 032 155 139 032 166 135 032 145 143 186
33260:169 007 141 021 208 173 030 208 076 212 140 173 002
33272:000 220 141 252 207 041 001 208 043 032 030 139 026
33284:240 003 032 186 138 173 253 207 201 030 208 003 142
33296:076 173 130 173 254 207 201 150 208 003 076 173 048
33308:130 238 254 207 206 253 207 173 252 207 141 249 241
33320:207 076 173 130 173 252 207 041 002 208 037 032 042
33332:030 139 240 003 032 186 138 173 253 207 201 200 062
33344:240 107 173 254 207 201 000 240 100 238 253 207 236
33356:206 254 207 173 252 207 141 249 207 076 173 130 047
33368:173 252 207 041 004 208 037 032 050 139 240 003 194
33380:032 186 138 173 253 207 201 030 240 063 173 254 002
33392:207 201 000 240 056 206 254 207 206 253 207 173 018
33404:252 207 141 249 207 076 173 130 173 252 207 041 184
33416:008 208 034 032 050 139 240 003 032 186 138 173 099
33428:253 207 201 200 240 019 173 254 207 201 150 240 189
33440:012 238 254 207 238 253 207 173 252 207 141 249 031
33452:207 032 155 139 162 255 160 000 200 208 253 232 127
33464:208 250 032 024 136 032 144 136 032 036 137 032 103
33476:029 135 032 030 139 208 008 032 050 139 208 003 185
33488:032 181 133 032 217 130 076 247 129 173 030 208 004
```




```
33500:041 001 208 001 096 206 199 207 104 104 032 200 083
33512:143 174 199 207 232 169 032 157 040 064 076 212 145
33524:140 173 241 207 010 141 207 207 176 008 169 000 131
33536:141 206 207 076 011 131 169 001 141 206 207 173 133
33548:207 207 024 105 013 141 014 208 173 206 207 105 086
33560:000 024 106 106 141 206 207 173 016 208 041 127 099
33572:013 206 207 141 016 208 173 240 207 024 105 041 081
33584:141 015 208 169 001 141 046 208 169 023 141 255 029
33596:067 162 254 154 173 021 208 141 205 207 169 129 158
33608:141 021 208 032 081 143 032 081 143 032 081 143 186
33620:169 000 141 202 207 169 004 141 203 207 173 203 111
33632:207 074 144 008 169 010 141 204 207 076 113 131 044
33644:169 020 141 204 207 173 203 207 141 245 207 169 146
33656:010 141 244 207 032 252 135 172 242 207 174 204 092
33668:207 204 240 207 208 008 236 241 207 208 003 076 129
33680:154 131 032 055 134 208 003 238 202 207 173 204 093
33692:207 024 105 020 201 160 240 010 201 150 240 006 184
33704:141 204 207 076 113 131 238 203 207 172 203 207 222
33716:192 020 208 166 160 000 185 195 132 032 210 255 143
33728:200 192 021 208 245 169 000 174 202 207 032 205 255
33740:189 169 032 032 210 255 169 042 032 210 255 169 176
33752:032 032 210 255 169 000 174 200 207 032 205 189 129
33764:169 032 032 210 255 169 061 032 210 255 169 032 062
33776:032 210 255 173 200 207 141 245 207 173 202 207 188
33788:141 244 207 032 252 135 174 242 207 173 243 207 205
33800:032 205 189 169 032 032 210 255 169 146 032 210 153
33812:255 173 200 207 201 070 240 006 024 105 005 141 111
33824:200 207 173 242 207 056 233 010 141 242 207 141 043
33836:221 207 173 243 207 233 000 141 243 207 013 221 105
33848:207 240 020 144 018 162 253 160 000 200 208 253 129
33860:232 208 250 162 011 032 035 134 076 034 132 234 072
33872:169 000 141 021 208 169 096 133 252 169 032 133 067
33884:254 160 000 133 251 133 253 177 253 145 251 200 254
33896:208 249 230 252 230 254 166 252 224 127 208 239 183
33908:177 253 145 251 200 192 064 208 247 169 007 141 122
33920:046 208 173 254 067 141 255 067 162 039 169 032 205
```

```
33932:157 120 064 202 224 007 208 248 160 006 185 216 145
33944:132 153 055 138 185 223 132 153 062 138 136 192 059
33956:255 208 239 032 145 143 173 205 207 056 042 141 218
33968:021 208 032 166 135 169 000 174 201 207 157 080 190
33980:064 238 201 207 076 212 140 159 019 017 017 017 019
33992:029 029 029 029 029 029 029 029 029 018 066 079 112
34004:078 085 083 032 010 030 040 060 080 110 130 040 222
34016:080 170 050 090 140 160 142 217 207 140 216 207 251
34028:200 032 055 134 201 003 240 093 232 032 055 134 111
34040:201 002 208 085 202 202 032 055 134 201 002 208 244
34052:076 173 216 207 024 105 019 168 032 055 134 201 134
34064:002 208 062 232 232 032 055 134 201 002 208 053 157
34076:173 217 207 024 105 009 170 173 216 207 024 105 122
34088:009 168 032 055 134 201 002 208 032 200 200 032 033
34100:055 134 201 002 208 023 173 217 207 056 233 009 034
34112:170 032 055 134 201 002 208 009 136 136 032 055 210
34124:134 201 002 240 001 096 174 217 207 172 216 207 151
34136:236 241 207 208 012 152 024 105 010 205 240 207 143
34148:208 003 076 245 130 162 011 032 035 134 169 003 028
34160:133 002 169 255 141 246 207 141 214 207 172 216 167
34172:207 200 238 246 207 173 246 207 201 019 240 044 048
34184:201 010 144 006 206 214 207 076 149 133 238 214 142
34196:207 173 217 207 024 109 214 207 141 215 207 173 194
34208:217 207 056 237 214 207 170 202 232 032 239 139 008
34220:236 215 207 208 247 076 125 133 096 172 253 207 043

34232:174 254 207 192 040 240 016 192 030 240 012 224 213
34244:150 240 008 152 056 233 020 168 032 230 132 172 253
34256:253 207 174 254 207 192 030 240 021 224 000 240 202
34268:017 224 010 240 013 152 056 233 010 168 138 056 001
34280:233 010 170 032 230 132 172 253 207 174 254 207 002
34292:192 030 240 021 224 150 240 017 224 140 240 013 183
34304:152 056 233 010 168 138 024 105 010 170 032 230 048
34316:132 172 253 207 174 254 207 192 200 240 011 192 198
34328:190 240 007 224 150 240 003 032 230 132 096 189 221
34340:000 064 201 057 240 004 254 000 064 096 169 048 209
34352:157 000 064 202 076 035 134 152 072 138 072 169 039
34364:096 133 252 169 000 133 251 138 072 074 074 170 086
34376:152 072 074 074 074 168 202 224 255 240 014 165 250
34388:251 024 105 008 133 251 144 242 230 252 076 078 086
34400:134 136 192 255 240 016 165 251 024 105 064 133 019
34412:251 165 252 105 001 133 252 076 097 134 104 041 183
34424:007 168 136 192 255 240 009 230 251 208 247 230 245
34436:252 076 122 134 104 041 003 170 169 192 141 247 247
34448:207 202 224 255 240 009 078 247 207 078 247 207 041
34460:076 145 134 200 173 247 207 049 251 141 221 207 159
34472:173 247 207 041 001 208 015 078 247 207 078 247 125
34484:207 078 221 207 078 221 207 076 168 134 104 170 003
34496:104 168 173 221 207 096 169 000 141 220 207 169 019
34508:255 141 219 207 174 220 207 232 236 219 207 240 201
34520:061 173 220 207 024 109 219 207 106 141 218 207 060
34532:141 245 207 141 244 207 032 252 135 173 242 207 150
34544:056 237 222 207 141 221 207 173 243 207 237 223 054
34556:207 013 221 207 240 026 144 009 173 218 207 141 010
34568:219 207 076 208 134 173 218 207 141 220 207 076 046
34580:208 134 173 220 207 141 218 207 096 173 241 207 197
34592:056 237 254 207 144 003 076 048 135 173 254 207 034
34604:056 237 241 207 141 245 207 141 244 207 032 252 206
34616:135 173 242 207 141 236 207 173 243 207 141 237 094
34628:207 173 240 207 056 237 253 207 144 003 076 088 167
34640:135 173 253 207 056 237 240 207 141 245 207 141 018
34652:244 207 032 252 135 173 242 207 024 109 236 207 112
34664:141 222 207 173 243 207 109 237 207 141 223 207 117
34676:032 198 134 173 218 207 074 074 074 170 168 169 015
34688:160 157 127 064 202 224 255 208 248 173 218 207 067
34700:041 007 170 189 158 135 153 128 064 169 032 153 003
34712:129 064 153 130 064 096 101 116 117 097 246 234 163
34724:231 160 173 027 212 041 015 201 013 176 247 024 148
34736:105 001 141 241 207 173 027 212 041 007 024 105 180
34748:001 141 245 207 169 020 141 244 207 032 252 135 190
34760:173 242 207 141 240 207 173 241 207 041 001 208 233
34772:009 173 240 207 056 233 010 141 240 207 173 241 094
34784:207 141 245 207 169 010 141 244 207 032 252 135 166
34796:173 242 207 141 241 207 173 240 207 024 105 030 178
34808:141 240 207 096 169 000 141 242 207 162 008 078 147
34820:245 207 144 004 024 109 244 207 106 110 242 207 061
34832:202 208 240 141 243 207 096 096 162 006 032 086 199
34844:136 208 049 032 106 136 208 044 173 027 212 201 024
34856:064 176 008 169 014 157 183 207 076 080 136 201 231
34868:128 176 008 169 013 157 183 207 076 080 136 201 050
34880:192 176 008 169 011 157 183 207 076 080 136 169 092
34892:007 157 183 207 202 224 255 208 197 096 189 055 008
34904:138 024 125 062 138 141 250 207 008 104 041 001 047
34916:141 251 207 076 085 139 189 055 138 056 253 062 216
34928:138 144 011 141 250 207 169 000 141 024 212 076 089
34940:085 139 189 062 138 056 253 055 138 141 250 207 045
34952:169 000 141 251 207 076 085 139 238 182 207 173 212
```




```
34964:182 207 205 181 207 240 001 096 169 255 141 182 166
34976:207 162 006 189 183 207 041 001 208 029 189 062 108
34988:138 201 030 208 003 076 030 137 189 055 138 201 042
35000:150 208 003 076 030 137 254 055 138 222 062 138 121
35012:076 030 137 189 183 207 041 002 208 023 189 062 007
35024:138 201 200 240 073 189 055 138 201 000 240 066 157
35036:254 062 138 222 055 138 076 030 137 189 183 207 119
35048:041 004 208 023 189 062 138 201 030 240 043 189 064
35060:055 138 201 000 240 036 222 055 138 222 062 138 215
35072:076 030 137 189 183 207 041 008 208 020 189 062 070
35084:138 201 200 240 013 189 055 138 201 150 240 006 247
35096:254 055 138 254 062 138 202 224 255 208 128 096 246
35108:162 006 169 128 141 247 207 189 055 138 010 176 128
35120:059 105 014 008 072 138 010 170 104 157 002 208 071
35132:138 074 170 040 173 247 207 073 255 045 016 208 170
35144:144 003 013 247 207 141 016 208 189 062 138 024 184
35156:105 041 072 138 010 170 104 157 003 208 138 074 024
```

```
35168:170 078 247 207 202 224 255 208 194 076 134 137 180
35180:024 105 014 072 138 010 170 104 157 002 208 138 226
35192:074 170 173 016 208 013 247 207 141 016 208 076 133
35204:080 137 206 246 207 208 035 169 010 141 246 207 232
35216:173 248 207 201 021 240 008 169 021 141 248 207 236
35228:076 164 137 169 020 141 248 207 162 006 157 249 100
35240:067 202 224 255 208 248 173 141 002 201 001 240 082
35252:249 096 162 006 169 007 157 040 208 169 020 157 084
35264:249 067 202 224 255 208 241 162 036 169 003 157 117
35276:120 216 202 224 255 208 248 162 039 169 032 157 188
35288:000 064 157 040 064 157 120 064 157 080 064 202 105
35300:224 255 208 239 160 000 185 069 138 032 210 255 155
35312:200 192 011 208 245 160 000 185 088 138 032 210 117
35324:255 200 192 043 208 245 160 007 185 080 138 153 070
35336:000 080 136 192 255 208 245 200 185 131 138 032 018
35348:210 255 200 192 040 208 245 160 000 185 170 138 231
35360:032 210 255 200 192 016 208 245 160 000 185 080 023
35372:142 153 000 069 200 208 247 173 030 208 096 010 044
35384:030 040 060 080 110 130 040 080 170 050 090 140 052
35396:160 158 019 017 017 017 083 079 078 065 082 058 133
35408:126 126 126 126 096 096 096 096 154 019 017 017 151
35420:076 069 086 069 076 058 032 032 032 032 032 032 206
35432:032 032 032 032 032 032 032 032 032 032 032 032 232
35444:032 032 032 032 032 032 032 032 032 032 032 032 244
35456:032 032 032 005 019 083 067 079 082 069 058 032 206
35468:048 048 048 048 048 048 032 032 032 032 032 032 108
35480:072 073 071 072 032 083 067 079 082 069 058 032 174
35492:048 048 048 048 048 048 153 019 017 083 084 065 105
35504:077 073 078 065 058 032 218 218 218 218 173 249 061
35516:207 041 001 208 009 238 254 207 206 253 207 076 047
35528:247 138 173 249 207 041 002 208 009 238 253 207 124
35540:206 254 207 076 247 138 173 249 207 041 004 208 174
35552:009 206 254 207 206 253 207 076 247 138 173 249 145
35564:207 041 008 208 006 238 254 207 238 253 207 032 087
35576:155 139 162 255 160 000 200 208 253 232 208 250 166
35588:032 024 136 032 144 136 032 036 137 032 217 130 068
35600:032 030 139 208 165 032 050 139 208 160 032 181 112
35612:133 096 173 254 207 024 109 253 207 141 250 207 034
35624:008 104 041 001 141 251 207 076 085 139 173 254 240
35636:207 056 237 253 207 144 011 141 250 207 169 000 142
35648:141 251 207 076 085 139 173 253 207 056 237 254 095
35660:207 141 250 207 169 000 141 251 207 173 250 207 231
35672:056 233 010 141 250 207 173 251 207 233 000-141 198
35684:251 207 048 028 173 250 207 013 251 207 240 020 203
35696:173 250 207 056 233 020 141 250 207 173 251 207 232
35708:233 000 141 251 207 076 102 139 096 169 002 133 137
35720:002 162 159 160 031 032 239 139 200 192 200 208 068
35732:248 202 224 150 208 241 096 173 254 207 010 176 033
35744:029 105 015 141 000 208 173 016 208 041 254 144 214
35756:002 009 001 141 016 208 173 253 207 024 105 041 072
35768:141 001 208 076 207 139 024 105 015 141 000 208 169
35780:173 016 208 009 001 141 016 208 076 178 139 169 250
35792:002 133 002 174 254 207 172 253 207 032 055 134 041
35804:201 001 208 005 162 012 032 035 134 174 254 207 109
35816:172 253 207 032 239 139 096 072 152 072 138 072 084
35828:169 096 133 252 169 000 133 251 138 072 074 074 013
35840:170 152 072 074 074 074 168 202 224 255 240 014 183
35852:165 251 024 105 008 133 251 144 242 230 252 076 101
35864:007 140 136 192 255 240 016 165 251 024 105 064 083
35876:133 251 165 252 105 001 133 252 076 026 140 104 138
35888:041 007 168 136 192 255 240 009 230 251 208 247 240
35900:230 252 076 051 140 104 041 003 170 169 063 141 220
35912:247 207 165 002 010 010 010 010 010 010 202 224 155
35924:255 240 012 074 074 056 110 247 207 110 247 207 131
35936:076 082 140 200 072 173 247 207 049 251 145 251 197
35948:104 017 251 145 251 104 170 104 168 104 096 173 003
35960:025 208 141 025 208 041 001 240 077 169 012 160 147
35972:059 162 024 142 022 208 141 033 208 140 017 208 216
35984:141 024 208 173 018 208 201 081 144 016 169 000 247
35996:141 018 208 173 017 208 041 127 141 017 208 076 251
36008:206 140 169 000 141 033 208 169 200 141 022 208 013
36020:169 027 141 017 208 169 004 141 024 208 169 081 002
36032:141 018 208 173 017 208 041 127 141 017 208 076 031
36044:049 234 104 168 104 170 104 064 169 070 141 254 043
36056:207 169 120 141 253 207 162 039 169 032 157 120 200
36068:064 202 224 007 208 248 032 155 139 032 029 135 167
36080:173 199 207 201 007 240 057 032 248 141 032 024 009
36092:136 032 144 136 032 036 137 162 255 160 000 200 146
36104:208 253 232 208 250 162 249 160 000 200 208 253 087
36116:232 208 250 173 000 220 041 016 208 220 162 013 227
36128:169 032 157 040 064 232 224 039 208 248 173 030 112
36140:208 076 247 129 162 039 169 013 157 040 216 202 166
36152:224 255 208 248 160 000 162 000 189 041 064 157 228
36164:040 064 232 224 037 208 245 185 037 142 141 077 164
36176:064 152 072 160 000 162 000 200 208 253 232 208 255
36188:250 104 168 173 000 220 041 016 240 010 200 192 170
36200:043 208 211 160 000 076 062 141 162 000 189 007 083
36212:064 221 031 064 240 018 144 021 162 005 189 007 002
36224:064 157 031 064 202 224 255 208 245 076 145 141 148
36236:232 224 006 208 225 162 005 189 031 064 157 192 043
36248:207 202 224 255 208 245 173 000 220 041 016 240 135
36260:249 032 073 145 032 182 137 162 005 189 192 207 233
36272:157 031 064 202 224 255 208 245 169 096 133 252 164
36284:169 032 133 254 160 000 133 251 133 253 177 253 088
36296:145 251 200 208 249 230 252 230 254 166 252 224 045
36308:127 208 239 177 253 145 251 200 192 064 208 247 219
36320:032 145 143 169 007 141 021 208 141 201 207 169 016
36332:040 141 200 207 169 012 141 199 207 076 212 140 188
36344:162 000 189 006 142 032 210 255 232 224 032 208 148
36356:245 096 005 019 017 029 029 029 029 029 029 029 077
36368:029 029 029 029 029 029 029 029 029 080 082 069 252
36380:083 083 032 066 085 084 084 079 078 135 129 141 083
36392:133 160 143 150 133 146 032 046 046 032 016 018 071
36404:005 019 019 032 002 021 020 020 015 014 032 020 015
36416:015 032 016 012 001 025 032 001 007 001 009 014 229
36428:032 046 046 032 000 000 000 000 000 000 000 000 232
36440:000 000 000 000 000 000 000 000 000 000 000 000 088
36452:000 000 048 000 000 048 000 000 252 000 000 252 188
36464:000 000 048 000 000 048 000 000 000 000 000 000 208
36476:000 000 000 000 000 000 000 000 000 000 000 000 124
36488:000 000 000 000 000 000 000 000 000 000 000 000 136
36500:000 000 000 000 000 000 000 000 000 000 000 000 148
36512:000 000 000 000 000 001 140 000 000 216 000 000 005
36524:112 000 000 112 000 000 216 000 001 140 000 000 241
36536:000 000 000 000 000 000 000 000 000 000 000 000 184
36548:000 000 000 000 000 000 000 000 000 000 000 000 196
36560:000 000 000 000 000 000 000 000 000 000 000 000 208
36572:000 000 000 000 000 000 000 000 000 000 096 000 060
36584:000 240 000 001 248 000 000 240 000 000 096 000 033
36596:000 000 000 000 000 000 000 000 000 000 000 000 244
```

```
36608:000 000 000 000 000 000 000 000 000 000 000 000 000
36620:000 000 000 000 000 000 000 000 000 000 000 000 012
36632:000 000 000 000 015 255 255 023 224 049 016 000 093
36644:001 035 102 051 102 219 051 075 108 103 199 254 056
36656:102 128 000 014 255 255 252 255 255 248 000 000 020
36668:000 000 000 000 000 000 000 000 000 000 000 000 060
36680:000 000 000 000 000 000 000 000 165 162 007 169 063
36692:000 157 000 212 202 224 255 208 248 169 141 141 249
36704:024 212 169 005 141 005 212 169 218 141 006 212 074
36716:169 150 141 001 212 169 139 141 000 212 169 017 092
36728:141 004 212 160 140 162 000 232 208 253 200 208 248
36740:250 169 016 141 004 212 232 208 253 200 208 250 227
36752:096 160 000 169 000 153 000 212 200 192 008 208 006
36764:246 169 143 141 024 212 169 008 141 005 212 169 003
36776:243 141 006 212 169 129 141 004 212 162 255 142 192
36788:001 212 202 160 255 136 192 001 208 251 224 001 231
36800:208 241 169 128 141 004 212 096 162 000 169 000 186
36812:157 000 212 232 224 008 208 248 169 143 141 024 178
36824:212 169 017 141 005 212 169 213 141 006 212 169 090
36836:002 141 003 212 169 100 141 002 212 169 005 141 245
36848:001 212 169 135 141 000 212 169 065 141 004 212 165
36860:169 252 160 000 162 000 200 208 253 232 208 250 042
36872:024 105 001 208 245 169 064 141 004 212 096 162 159
36884:000 142 033 208 142 032 208 189 206 144 240 007 035
36896:032 210 255 232 076 027 144 234 173 190 207 105 125
36908:048 032 210 255 169 157 032 210 255 032 228 255 135
36920:208 010 173 000 220 041 016 208 244 076 091 144 207
36932:201 013 240 019 201 049 144 233 201 058 176 229 040
36944:072 056 233 048 141 190 207 104 032 210 255 162 254
36956:000 160 000 232 -208 253 200 208 250 173 000 220 204
36968:041 016 240 249 189 050 145 240 007 032 210 255 242
36980:232 076 108 144 173 180 207 032 210 255 169 157 011
36992:032 210 255 032 228 255 208 010 173 000 220 041 000
37004:016 208 244 076 167 144 201 013 240 017 201 069 200
37016:240 007 201 072 240 003 076 131 144 141 180 207 002
37028:032 210 255 234 056 169 010 237 190 207 010 073 055
37040:255 024 105 002 141 177 130 141 251 138 141 004 149
37052:141 173 180 207 056 233 069 074 073 001 141 181 181
37064:207 169 001 133 204 096 014 147 017 017 159 018 102
37076:029 029 029 029 029 029 029 032 032 032 032 032 063
37088:032 032 017 157 157 157 157 157 157 157 032 211 111
37100:208 201 203 197 032 146 032 045 032 005 032 194 027
37112:089 032 211 072 065 078 069 032 211 084 069 086 066
37124:069 078 083 013 029 029 029 029 029 029 029 159 097
37136:018 032 032 032 032 032 032 032 013 013 013 013 054
37148:013 029 029 029 029 029 029 211 080 069 069 068 200
37160:032 040 049 045 057 041 063 032 159 000 013 013 072
37172:013 029 029 029 029 029 029 158 197 065 083 089 063
37184:047 200 065 082 068 063 032 159 000 120 173 013 062
37196:220 009 129 141 013 220 169 000 141 026 208 169 241
37208:234 141 021 003 169 049 141 020 003 088 169 021 123
37220:141 024 208 169 027 141 017 208 169 199 141 000 008
37232:221 169 004 141 136 002 169 000 141 021 208 032 076
37244:019 144 169 064 141 136 002 169 198 141 000 221 248
37256:169 008 141 024 208 169 216 133 252 160 000 132 212
37268:251 169 006 145 251 200 208 251 230 252 166 252 225
37280:224 220 208 243 032 183 128 169 255 141 182 207 048
37292:096 253 208 068 069 131 000 000 000 000 083 000 056
```

# BUILD A BETTER BASIC

**Nick Hampshire brings you some more commands to improve your Basic.**

IN THE LAST FOUR ARTICLES in this series I have given all the initialisation and wedge routines needed to add extra commands to the Basic of a C64 computer. I have also given the code to add 12 new commands which are; CTL, APPEND, CHANGE, DUMP, FIND, AUTO, CHAIN, DELETE, RENUMBER, MAT, SORT and VARPTR.

This month I am adding a further eight commands. These are, CATALOG, DISK, EXEC, MERGE, GET, PUT, TYPE and OLD. Seven of the eight are special disk control commands, and eight (OLD) is included since it is called by the other routines. These disk control commands add some very powerful and useful features to a disk based 64 system and will save a considerable amount of programming time.

All eight new commands require that the wedge and initialisation code given in the first issue are present in memory at the correct locations, and that their command names and entry points are stored in the correct tables. These eight commands are independent of all the previously added commands – except APPEND – routines from which are required by the new routines. Within this limitation they can be used without the previously added code. To ensure that you have the wedges and new routines correctly positioned. The Basic loader at the end of this article gives the initialisation routines and all commands.

In next month's issue I will show how to write and add your own commands to Basic. All the programs used in this series are extracted from the book *Advanced Commodore 64 Basic Revealed* by Nick Hampshire and published by Collins.

## GET

**Abbreviated entry:** G(shift)E
**Affected Basic abbreviations:** None
**Token:** Hex $EE, $OF, Decimal 238,15
**Modes:** Direct and program
**Recommended mode:** Either; different effects in direct mode and program mode
**Purpose:** To input an ASCII file on disk into memory with line numbers created from 1000 in steps of 10. GET will read in files created by the Commodore assembler and SYSRES. Each time is read in until a carriage return is reached. It is then tokenised and entered into memory as a program line.
**Syntax:** Direct mode; GET filename, d – where di is the device number (disk only)

Run mode: as chapter 7 GET and GET #.

**Errors:** Illegal device – if the device number specified is less than eight Missing file

Missing file name – if a null filename is specified.

File not found – if the file does not exist

Device not present – connected

File open error – if 10 files are already open

Disk errors – at the end, the disk error channel is read and displayed.

**Use:** For editing Commodore assembler files or files for the use of the EXEC command
**Routine entry point:** $8ED1
**Routine operation:** The GET routine first checks whether the computer is in run mode or direct. If it is in run mode, then the Basic version of GET is performed. If in direct mode, the file parameters are read in and checked for a null filename or the device not being disk. If these checks are OK, the message 'reading' filename is displayed and the file is opened. Each line is then input and stored in the input buffer, tokenised, and entered into memory until the end of file marker is reached. The program is then re-chained and the variable pointers are sest to the correct values for the program. Finally the disk error channel is rad and displayed.

**GET**

```
1000 GET LDA $9D ;CHECK IF DIRECT
1010  BNE GETUN ;YES, DIRECT
1020  JSR $0079 ;GET CURRENT CHAR
1030  JMP $AB7E ;PERFORM BASIC 'GET'
1040 GETUN JSR DPARS ;GET FILE PARAMETERS
1050  JSR GETMES :'READING'
1060  JSR GETOPN ;OPEN FILE
1070  JSR GETIN ;SET INPUT
1080  LDA $2B :SET START OF PROGRAM
1090  STA $FB ;POINTER
1100  LDA $2C
1110  STA $FC
1120  LDA $2B
1130  CLC
1140  ADC #$02
1150  TAX
1160  LDA $2C
1170  ADC #$00
1180  STA $2E
1190  STA $30
1200  STA $32
1210  STX $2D
1220  STX $2F
1230  STX $31
1240  LDA #$03 ;START LINE# HI
1250  LDX #$E8 ;START LINE# HI
1260  STA GETLNO+1
1270  STX GETLNO
1280 GETLP1 LDY #$00
1290 GETLP2 JSR $FFCF ;INPUT BYTE
1300  CMP #$0D ;END OF LINE?
```

```
 BEQ GETLN ;YES
 CMP #$0A ;LINE FEED?
 BEQ GETLP2 ;YES
 STA $0200,Y ;STORE BYTE
 INY
 CPY #$57 ;END OF BUFFER?
 BNE GETLP2
GETLN LDA $90 ;STATUS
 STA GETER
 LDA #$00 ;TERMINATOR
 STA $0200,Y ;STORE
 LDX #$00
 STX $7A
 LDA #$02
 STA $7B
 JSR $A579 ;CRUNCH LINE
 LDA $0200
 BEQ GETLP4 ;NULL LINE
 LDY #$02
 LDA GETLNO ;LINE# LO
 STA ($FB),Y ;STORE IT
 INY
 LDA GETLNO+1 ;LINE# HI
 STA ($FB),Y ;STORE IT
GETLP3 INY
 LDA $01FC,Y ;GET BYTE
 STA ($FB),Y ;STORE IT
 BNE GETLP3 ;UNTIL END OF LINE
 INY
 TYA
 LDY #$00
 CLC
 ADC $FB ;INCREASE POINTER BY
 STA $FD ;LENGTH
 STA ($FB),Y
 LDA $FC
 ADC #$00
 INY
 STA ($FB),Y
 TAY
 LDA $FD
 STA $FB
 STY $FC
 TYA
 BMI GETEND
 LDA GETLNO ;INCREASE LINE#
 CLC
 ADC #$0A ;BY 10
 STA GETLNO
 LDA GETLNO+1
 ADC #$00
 STA GETLNO+1
GETLP4 LDA GETER ;STATUS?
 BNE GETEND ;BAD
 JMP GETLP1 ;DO NEXT LINE
;
GETEND LDA #$00
 TAY
 STA ($FB),Y ;ZERO END OF PROGRAM
 INY
 STA ($FB),Y
 JSR PUTEND ;CLOSE AND DISK
 JSR OLD ;RESET POINTERS
 JMP $A474
GETMES LDA #<GMESSG ;POINTER TO
 LDY #>GMESSG ;'READING'
 JSR $AB1E ;PRINT STRING
 JMP $F5C1 ;PRINT FILENAME
GMESSG .BYT 'READING ',$00
GETIN LDX $B8
 JSR $FFC6 ;SET INPUT
 BCS GETIN1 ;ERROR
 RTS
GETIN1 JMP $E0F9 ;SEND ERROR
;
GETOPN LDY #$00
GETOP1 LDA ($BB),Y ;GET BYTE
 STA $0200,Y ;STORE IT
 INY
 CPY $B7 ;END OF FILENAME?
 BNE GETOP1 ;NOT YET
 LDX #$00
GETOP2 LDA GETSR,X ;GET BYTE
 STA $0200,Y ;STORE IT
 INX
 INY
 CPX #$04 ;END OF SR?
 BNE GETOP2 ;NOT YET
 LDA #$60
 STA $B9
 JMP PUTOP4 ;COMPLETE OPEN
GETSR .BYT ',S,R'
GETLNO .WOR 0
GETER .BYT 0
 .END
```

## CATALOG

**Abbreviation**: C(shift)A
**Affected Basic abbreviations:** NONE
**Token:** Hex $EE,$05, Decimal 238,5
**Modes:** Direct and program
**Recommended mode:** Direct
**Purpose:** To display the directory (CATALOG) of a disk in drive unit EIGHT. This comand will display the directory straight to the screen without having to load it in. Users of dual disk drives will be pleased to note that you can specify which drive to display by either a number one or zero after the command. If no number is specified, the routine will default to drive zero.
**Syntax:** CATALOG [0 or 1]
**Errors:** Syntax error – if the command CATALOG is followed by anything but '0,1,:' or nothing.
Disk error message – after the CATALOG has been displayed, the disk error channel is read and displayed.
**Use:** The command is used to display the directory of a disk. This can be useful if you have a program that you wish to save but need to check if there is room on the disk or find a filename to use. The directory can be paused when displaying, by use of the spacebar, and restarted with any key. Display can be stopped completely with the STOP key.
**Routine entry point:** $85B6
**Routine operation:** On entry, the routine checks to see if a drive number is specified. If no number is specified or zero, the character '0' is inserted into the filename after the '$'. If it is a one, the character '1' is inserted. Anything else will cause syntax error. The file is then opened and each line is read and displayed ignoring line links. When the directory is finished, the file is closed and the disk error channel is read. Check is the one following str2.

### CATALOGUE

```
CATLOG BEQ CATL01 ;DRIVE 0
 CMP #$30 ;IS IT 0?
 BEQ CATL01 ;YES
 CMP #$31 ;IS IT 1?
 BEQ CATL02 ;YES
 JMP $AF08 ;SYNTAX ERROR
CATL01 LDA #$30 ;CHAR '0'
 .BYT $2C
CATL02 LDA #$31 ;CHAR '1'
 STA OPDIR+1 ;STORE IN STRING
 LDA #$02 ;LENGTH
 LDX #<OPDIR ;ADDRESS LSB
 LDY #>OPDIR ;MSB
 JSR $FFBD ;SET FILENAME DETAILS
 LDA #$0E
 JSR GETN1 ;GET UNUSED FILE#
 LDX #$08 ;DEVICE 8
 LDY #$00 ;SA 0
 JSR $FFBA ;SET FILE DETAILS
 JSR $FFC0 ;OPEN FILE
 BCC CATL03 ;NO ERROR
 PHA ;STORE ERROR
 LDA $B8 ;GET FILE #
 JSR $FFC3 ;CLOSE FILE
 PLA ;GET ERROR
 JMP $E0F9 ;SEND ERROR
:
CATL03 LDY #$03
CATL04 STY $B7
 LDX $B8
 JSR $FFC6 ;SET INPUT DEVICE
 JSR $FFCF ;INPUT
 STA $57 ;STORE VALUE
```

## DISK

**Abbreviated entry:** D(shift)I
**Affected Basic abbreviations:** DIM - DIM
**Token:** Hex $EE,$0A, Decimal 238,10
**Modes:** Direct and program
**Recommended mode:** Either
**Purpose:** To send a disk command to the disk unit eight.
**Syntax:** DISK [string expression] –where the string expression is:

"S0:TEST" – to scratch the file test.
"N0:DISK,00" – to reformat the entire disk.

The other Syntax is DISK which will display the disk error message to the screen giving a message like:

23, READ ERROR,18,01

where 23 is the error number, 18 is the track, 01 is the sector, and READ ERROR is the error description.
**Errors:** Syntax error – if the first character of the command is not a quote character.
String too long – if the command is over 255 bytes long.
Type mismatch – if the command is a number, not a string.
**Use:** This command is useful in checking errors created from disk access by using just DISK which displays the message. A Basic equivalent would be:

OPEN 15,8,15
INPUT # 15,E,EM$,T,S
PRINT E", "EM$", "T", "S","
CLOSE 15

Also, for sending disk commands such as scratch a file etc:

DISK "10"
is equivalent to:

OPEN 15,8,15,"10"

For disk commands, refer to the disk user manual.
**Routine entry point:** $8A4D
**Routine operation:** The DISK routine checks to see if anything follows the command; if not the error channel is read and displayed. If there is text after the command (which must start with the quotes character) the text is read in and sent in the open command. Before either of these two operations are actioned, the current file is closed.

```
1330  JSR $FFB7 ;GET STATUS
1340  BNE CATL13 ;STATUS ERROR
1350  JSR $FFCF ;INPUT
1360  STA $57+1 ;STORE IT
1370  JSR $FFB7 ;GET STATUS
1380  BNE CATL13 ;STATUS ERROR
1390  LDY $B7 ;GET COUNTER
1400  DEY ;DO NEXT
1410 CATL05 BNE CATL04
1420  STY $B7 ;SET $B7 TO ZERO
1430 CATL06 JSR $FFCF ;INPUT
1440  PHA ;STORE IT
1450  JSR $FFB7 ;GET STATUS
1460  TAX ;STORE TO X
1470  PLA ;GET INPUT CHAR
1480  CPX #$00 ;WAS THERE AN ERROR?
1490  BNE CATL13 ;YES
1500  LDY $B7 ;GET LENGTH
1510  CPY #$50 ;TOO LONG?
1520  BCS CATL13 ;YES, ERROR
1530  STA $0200,Y ;STORE CHARACTER
1540  TAX
1550  BEQ CATL07 ;END OF LINE
1560  INC $B7 ;DO NEXT CHAR
1570  BNE CATL06 ;ALWAYS
1580 ;
1590 CATL07 JSR $FFCC ;RESET DEFAULT IO
1600  LDX $9F
1610  CPX #$03
1620  BEQ CATL08
1630  LDX $9E
1640  JSR $FFC9 ;SET OUTPUT DEVICE
1650 CATL08 LDX $57
1660  LDA $57+1
1670  JSR $BDCD ;PRINT FILE LENGTH
1680  LDA #$20 ;SPACE CHAR
1690  JSR $FFD2 ;PRINT IT
1700  LDY #$00
1710 CATL09 LDA $0200,Y ;GET CHAR
1720  BEQ CATL10 ;END OF LINE
1730  JSR $FFD2 ;PRINT CHAR
1740  INY
1750  BNE CATL09 ;DO NEXT LINE
1760 CATL10 LDA #$0D ;CARRIAGE RETURN
1770  JSR $FFD2 ;PRINT IT
1780  JSR $FFCC ;RESET DEFAULT IO
1790  JSR $FFE1 ;STOP KEY?
1800  BEQ CATL13 ;YES
1810  JSR $FFE4 ;GET KEY
1820  CMP #$20 ;SPACE?
1830  BNE CATL12 ;NO
1840 CATL11 JSR $FFE4 ;GET KEY
1850  BEQ CATL11 ;NO KEY
1860 CATL12 LDY #$02
1870  BNE CATL05 ;DO NEXT LINE
1880 CATL13 JSR $FFCC ;RESET DEFAULT IO
1890  LDA $B8 ;GET FILE NUMBER
1900  JSR $FFC3 ;CLOSE FILE
1910  JSR DISK01
1920  JMP $A474 ;JUMP TO READY VIA ERROR
1930 OPDIR .BYT '$0' ;FILE OPEN NAME
1940 .END
```

### DISK

```
1000 DISK JSR $0079 ; CHECK FOR BLANK
1010  BEQ DISK01 ; AFTER COMMAND.
1020  JMP DISK04
1030 DISK01 LDA #$00 ; IF BLANK, READ
1040  STA $B7 ; ERROR MESSAGE
1050  JSR FOPEN ; OPEN A FILE
1060  LDA #$0D ; PRINT <RETURN>
1070  JSR $FFD2
1080  LDA #$12 ; PRINT <REVERSE ON>
1090  JSR $FFD2
1100  LDX $B8
1110  JSR $FFC6 ; SET FILE TO INPUT
1120 DISK02 JSR $FFCF ; INPUT
1130  PHA
1140  LDA $90 ; CHECK STATUS
1150  BNE DISK03
1160  PLA
1170  JSR $FFD2 ; PRINT CHARACTER
1180  JMP DISK02 ; AND NEXT
1190 DISK03 PLA
1200  LDA $B8
1210  STA $49
1220  JSR $E1CC ; CLOSE FILE
1230  LDA #$92
1240  JSR $FFD2 ; PRINT <REVERSE OFF>
1250  LDA #$0D
1260  JSR $FFD2 ; PRINT <RETURN>
1270  LDA #$00
1280  JMP $FFC6 ; INPUT TO KYBD
1290 ;
1300 FOPEN JSR GETFNO ; FIND FREE FILE NO.
1310  STA $B8
1320  LDA #$0F ; SECONDARY ADDRESS
1330  STA $B9
1340  LDA #$08 ; DEVICE NUMBER
1350  STA $BA
1360  JMP $E1C1 ; OPEN
1370 ;
1380 GETFNO LDA #$0F ; CHECK TABLE OF
1390 GETN1 LDX $98 ; FILE NUMBERS FOR
1400  CPX #$00 ; A FREE ONE
1410  BEQ GETN4 ; HAS BEEN FOUND
1420 GETN2 CMP $0258,X
1430  BNE GETN3
1440  SEC
1450  SBC #$01
1460  JMP GETN1
1470 GETN3 DEX ; TRY NEXT NUMBER
```

```
1480  BNE GETN2
1490 GETN4 RTS
1500 ;
1510 DISK04 CMP #$22 ; CHECK  FOR COMMAND
1520  BEQ DISK05 ; IN QUOTES
1530  JMP $AF08 ; SYNTAX ERROR
1540 DISK05 LDA $B8 ; CLOSE CURRENT
1550  STA $49 ; DISK FILE
1560  JSR $E1CC
1570  JSR $AD9E ; GET TEXT IN QUOTES
1580  JSR $B6A3
1590  LDX $22 ; STRING ADDRESS AT
1600  STX $BB ; ($22)
1610  LDY $23
1620  STY $BC
1630 DISK07 STA $B7 ; SET LENGTH
1640  JSR FOPEN ; OPEN FILE
1650  LDA #$0D
1660  JSR $FFD2 ; PRINT <RETURN>
1670  RTS ; EXIT DISK
1680 .END
```

## EXEC

```
1000 EXEC JSR DPARS ;GET FILE PARAMETERS
1010  JSR GETOPN ;OPEN FILE
1020  LDA #$93 ;CLEAR SCREEN
1030  JSR $FFD2
1040  LDA $0300 ;STORE OFF ERROR LINK
1050  STA EXECER
1060  LDA $0301
1070  STA EXECER+1
1080  LDA $0302 ;STORE OFF WARM START
1090  STA EXECST
1100  LDA $0303
1110  STA EXECST+1
1120  LDA #<MERGRT ;SET 'RESET INPUT'
1130  STA $032C ; TO RTS
1140  LDA #>MERGRT
1150  STA $032D
1160  LDA #<EXEC06 ;SET ERROR VECTOR
1170  STA $0300
1180  LDA #>EXEC06
1190  STA $0301
1200  LDA #<EXEC02 ;SET WARM START
1210  STA $0302
1220  LDA #>EXEC02
1230  STA $0303
1240 EXEC02 LDX EXECNO
1250  JSR $FFC6 ;SET INPUT
1260  LDX #24 ;BOTTOM
1270  LDY #$00 ; LEFT
1280  CLC
1290  JSR $FFF0 ; OF SCREEN
1300  LDX #$00
1310 EXEC03 JSR $FFCF ;GET BYTE
1320  PHA
1330  LDA $90 ;CHECK STATUS
1340  BNE EXEC05
1350  PLA
1360  CMP #$0D ;CARRIAGE RETURN?
1370  BEQ EXEC04
1380  STA $0200,X
1390  INX
1400  JSR $FFD2 ;PRINT CHAR
1410  JMP EXEC03
1420 EXEC04 LDA #$00
1430  STA $0200,X
1440  LDA #$01
1450  STA $C6
1460  LDA #$0D
1470  JSR $FFD2
1480  LDX #$00 ;SET KEYBOARD AS INPUT
1490  JSR $FFC6
1500  LDX #$FF
1510  LDY #$01
1520  JMP $A486 ;EXEC IT
1530 EXEC05 JSR EXEC07 ;RESET VECTORS
1540  JSR DISK01 ;DISPLAY DISK ERROR
1550  JMP $A474 ;EXIT TO READY
1560 EXEC06 BCC EXEC02
1570  TXA ;SAVE ERROR NUMBER
1580  PHA
1590  JSR EXEC07 ;RESET VECTORS
1600  PLA ;RESTORE ERROR NUMBER
1610  TAX
1620  JMP ($0300) ;SEND ERROR
1630 EXEC07 LDA #$2F ;RESTORE
'RESET DEFAULT IO'
1640  STA $032C
1650  LDA #$F3
1660  STA $032D
1670  LDA EXECER ;RESET ERROR LINK
1680  STA $0300
1690  LDA EXECER+1
1700  STA $0301
1710  LDA EXECST ;RESET WARM START
1720  STA $0302
1730  LDA EXECST+1
1740  STA $0303
1750  LDX EXECNO
1760  JSR $FFC3 ;CLOSE FILE
1770  RTS
1780 EXECST .WOR 0
1790 EXECER .WOR 0
1800 EXECNO .BYT 0
1810 .END
```

## EXEC

**Abbreviated entry:** E(shift)X
**Affected Basic abbreviations:** EXP – EXP
**Token:** Hex $EE,$0D, Decimal 238,13
**Modes:** Direct and program
**Recommended mode:** Direct only
**Purpose:** To EXECute a text file stored on disk. This command works in conjunction with GET and PUT.
**Syntax:** EXEC filename,d – where d is the device number (disk only).
**Errors:** Illegal device – if the device number specified is less than eight.
Missing filename – if a null filename is specified
File not found – if the file does not exist
Device not present – if no disk drive is connected
Too many files – if 10 files are already open
Disk errors – at the end, the disk error channel is read and displayed.

**Use:** EXEC can be used in several different ways. The main one is to set up function keys when first powered up. For example enter the program:

```
10 CTL(,,5,0,0,1)
20 KEY1,"CATALOG"+CHR$(13)
30 KEY2,"DISK"+CHR$(13)
40 KEY3,"LIST'+CHR$(13)
50 KEY4,"RUN"+CHR$(13)
60 KEY5,"OLD"+CHR$(13)
70 KEY6,"PEEK("
80 KEY7,"RENUMBER"
90 KEY8,"FIND"
100 PRINT CTL(12,12,,,,1)"FUNCTION KEYS DEFINED"
```

Use the PUT command to write this to a disk file: PUT"FK",8

When powered up, type EXEC"FK",8 and the commands will be carried out and your function keys will be defined.

Other uses could be a string of change commands to a program.

**Routine entry point:** $8CCE
**Routine operation:** The filename and device number are read in and the file is opened. Each line is read into the input buffer until carriage return is found. It is then tokenised, and executed until the file is complete or an operating error occurs.

## MERGE

**Abbreviated entry:** M(shift)E
**Affected Basic abbreviations:** None
**Token:** Hex $EE,$12, Decimal 238,18
**Modes:** Direct and program
**Recommended mode:** Direct only
**Purpose:** To merge a Basic program from disk into the current Basic program in memory.
**Syntax:** MERGE filename,d – where d is the device number (disk only).
**Errors:** Illegal device – if the device number specified is less than eight.
Missing filename – if a null filename is specified.
File not found – if the file does not exist.
Device not present – if no disk drive is connected.
File open error – if 10 files are already open.
Disk errors – at the end, the disk error channel is read and displayed.
**Use:** Merge is used to combine two Basic programs in memory. Each line of the program on disk is read in until the zero byte is reached, and stored in the input buffer. The Basic routine to enter a line is then called and the line is entered at the correct place. Note: If a line number of the program to MERGE is the same as an existing line number, the MERGEd line will replace it.
**Routine entry point:** $97B0
**Routine operation:** The filename and device are read in and checked for missing filename and illegal device. If both checks are OK, the file is opened and the message MERGING is displayed. Each line is then read into the input buffer and entered using the Basic routine to do so. When the file is completed, it is closed and the disk error channel is read and displayed.

### MERGE

```
1000 MERGE JSR DPARS ; GET FILE PARAMETERS
1010  LDA #<MRGMES ; DISPLAY MERGE MESSAGE
1020  LDY #>MRGMES
1030  JSR $AB1E
1040  JSR $F5C1 ; DISPLAY FILENAME
1050  LDA $0302 ; SAVE BASIC WARM START
1060  STA MERGST ; LINK
1070  LDA $0303
1080  STA MERGST+1
1090  LDA #$0E ; FIND FILE NUMBER
1100  JSR GETN1
1110  STA $B8
1120  STA FILENO
1130  LDA #$00
1140  STA $B9
1150  JSR $FFC0 ; OPEN FILE
1160  LDX FILENO
1170  JSR $FFC6 ; SET FILE TO INPUT
1180  LDA #<MERGRT
1190  STA $032C
1200  LDA #>MERGRT ; SET 'RESET INPUT'
1210  STA $032D ; TO A RTS
1220  LDA #<MERG04
1230  STA $0302
1240  LDA #>MERG04 ; SET BASIC WARM START
1250  STA $0303 ; TO MERG04
1260  JSR $FFCF ; INPUT 2 BYTE LOAD
1270  JSR $FFCF ; ADDRESS
1280 MERG02 JSR $FFCF ; INPUT NEXT LINE
1290  STA $14 ; POINTERS AND
1300  JSR $FFCF ; CHECK FOR ZERO
1310  STA $15 ; (END OF BASIC PROGRAM)
1320  ORA $14
1330  BEQ MERG05
1340  LDA $90 ; CHECK STATUS
1350  BNE MERG05
1360  JSR $FFCF ; INPUT LINE NUMBER
1370  STA $14 ; AND STORE IN $14 & $15
1380  JSR $FFCF
1390  STA $15
1400  LDY #$00
1410 MERG03 JSR $FFCF ; INPUT LINE AND
1420  STA $0200,Y ; STORE IN INPUT
1430  LDX $C5 ; BUFFER
1440  CPX #63
1450  BEQ MERG05
1460  INY
1470  CMP #$00
1480  BNE MERG03 ; END OF LINE? NO.
1490  TYA ; YES
1500  CLC
1510  ADC #$04
1520  STA $0B
1530  LDA $90 ; CHECK STATUS
1540  BNE MERG05
1550  LDY $0B
1560  JMP $A4A4 ; MERGE LINE
1570 MERG04 JMP MERG02 ; DO NEXT LINE
1580 MERG05 LDA MERGST ; RESET BASIC WARM
1590  STA $0302 ; START
1600  LDA MERGST+1
1610  STA $0303
1620  LDA #$2F ; AND 'RESET DEFAULT I/O'
1630  STA $032C
1640  LDA #$F3
1650  STA $032D
1660  LDA FILENO
1670  JSR $FFC3 ; CLOSE FILE
1680  JSR $FFCC ; RESET DEFAULT I/O
1690  JSR DISK01 ; DISPLAY ERROR CHANNEL
1700  JMP $A474 ; JUMP TO READY
1710 MERGRT RTS
1720 FILENO .BYT 0
1730 MRGMES .BYT $91,'MERGING: ',$00
1740 MERGST .WOR 0
1750 ;
1760 ;GET PARAMETERS AND CHECK FOR
1770 ;ILLEGAL DEVICE. USED BY DISK
1780 ;ONLY COMMANDS.
1790 ;
1800 DPARS JSR $E1D4 ;GET FILENAME ETC
1810  LDA $BA ;IS DEVICE DISK?
1820  CMP #$08
1830  BCC PARERR ;NO
1840  LDA $B7 ;FILENAME LENGTH
1850  BEQ PARER1 ;ZERO
1860  RTS
1870 PARERR LDX #$09 ;ILLEGAL DEVICE
1880  .BYT $2C
1890 PARER1 LDX #$08 ;MISSING FILENAME
1900  JMP $A437 ;SEND ERROR
1910 .END
```

## OLD

**Abbreviated entry:** O(shift)L
**Affected Basic abbreviations:** None
**Token:** Hex $EE, $13, Decimal 238,19
**Modes:** Direct and program
**Recommended mode:** Direct only (there should be no program in memory)
**Purpose:** To restore a Basic program after a NEW has been performed.
**Syntax:** OLD
**Errors:** None
**Use:** OLD can be used if the program in memory has been wiped out using the NEW command. OLD will not work if DELETE was used to remove the whole program or if a variable has been declared since the NEW. (In most cases, syntax error will create a variable e.g. LI instead of L(shift)I will create the variable LI and give syntax error instead of trying to list the program).
**Routine entry point:** $9885
**Routing operation:** The first line is scanned until the end and the pointer to the next line is restored. The program is then re-chained and variable pointers are set.

## PUT

Abbreviated entry: P(Shift)U
**Affected Basic abbreviations:** None
**Token:** Hex $EE,$15, Decimal 238,21
**Modes:** Direct and program
**Recommended mode:** Direct
**Purpose:** To list a Basic program to a disk file without line numbers
**Syntax:** PUT filename,d – where d is the device number (disk only)
**Errors:** Illegal device – if the device number specified is less than eight.
Missing filename – if a null filename is specified.

## OLD

```
1000 OLD LDA $2B ; FIND THE END OF
1010  CLC ; THE FIRST LINE
1020  ADC #$04
1030  STA $57 ; SET POINTER TO AFTER
1040  LDA $2B ; LINE NUMBER
1050  ADC #$00
1060  STA $57+1
1070  LDY #$00
1080 OLD01 LDA ($57),Y ; SEARCH LINE
1090  BEQ OLD02 ; IF ZERO, END OF LINE
1100  LDA $57
1110  CLC
1120  ADC #$01 ; INCREMENT POINTER
1130  STA $57
1140  LDA $57+1
1150  ADC #$00
1160  STA $57+1
1170  JMP OLD01
1180 OLD02 LDA $57 ; END OF LINE
1190  LDY #$00 ; FOUND
1200  CLC
1210  ADC #$01
1220  STA ($2B),Y ; SET NEXT LINE
1230  INY ; POINTER
1240  LDA $57+1
1250  ADC #$00
1260  STA ($2B),Y
1270  JMP RESVAR ; SET VARIABLE POINTERS
1280 .END
```

## PUT

```
1000 PUT JSR DPARS ;GET FILENAME PARAMETERS
1010  JSR PUTMES ;'WRITING..'
1020  JSR PUTOPN ;OPEN FILE
1030  JSR PUTOUT ;SET OUTPUT
1040  JSR $A533 ;RE-CHAIN PROGRAM
1050  LDA $2B ;SET PROG POINTER
1060  STA $5F ;TO START OF PROGRAM
1070  LDA $2C
1080  STA $60
1090 PUT02 LDY #$00 ;END OF PROGRAM?
1100  LDA ($5F),Y
1110  INY
1120  ORA ($5F),Y
1130  BEQ PUTEND ;YES
1140  LDY #$04 ;POINT TO FIRST CHAR
1150 PUT03 LDA ($5F),Y
1160  BEQ PUTNL ;END OF LINE
1170  BMI PUTTK ;PRINT TOKEN
1180  CMP #$22 ;IS IT A QUOTE?
1190  BEQ PUTQT ;YES DO IT
1200 PUT04 JSR $FFD2 ;PRINT CHAR
1210  INY ;SET TO NEXT
1220  BNE PUT03 ;DO NEXT (ALWAYS)
1230 ;
1240 PUTEND LDA #$0D ;CARRIAGE RETURN
1250  JSR $FFD2 ;PRINT IT
1260  JSR PUTCLS ;CLOSE FILE
1270  JMP DISK01 ;DISPLAY DISK MESSAGE
1280 PUTNL LDY #$00
1290  LDA ($5F),Y ;GET LINK LO
1300  TAX
1310  INY
1320  LDA ($5F),Y ;GET LINK HI
1330  STA $60 ;STORE AS NEXT POINTER
1340  STX $5F
1350  LDA #$0D ;CARRIAGE RETURN
1360  JSR $FFD2 ;PRINT IT
1370  LDA $90 ;STATUS
1380  BNE PUTEND ;EXIT IF BAD
1390  JMP PUT02
1400 ;
1410 PUTQT JSR $FFD2 ;PRINT IT
1420  INY ;NEXT BYTE
1430  LDA ($5F),Y ;GET BYTE
1440  BEQ PUTNL ;END OF LINE
1450  CMP #$22 ;QUOTE?
1460  BNE PUTQT ;NO
1470  BEQ PUT04 ;OUTPUT AND DO NEXT
1480 ;
1490 PUTTK CMP #$EE ;MY TOKEN?
1500  BEQ PUTTK1 ;YES
1510  JSR PRIN09 ;TOKEN TO TEXT
1520  BMI PUTTK2 ;ALWAYS
1530 PUTTK1 JSR PRIN03 ;CONVERT TO
TEXT AND PRINT
1540 PUTTK2 AND #$7F ;MASK TOP BIT
1550  LDY $49 ;RESTORE .Y
1560  JMP PUT04 ;SEND AND DO NEXT
1570 ;
1580 PUTOUT LDX $B8 ;FILE NUMBER
1590  JSR $FFC9 ;SET OUTPUT
1600  BCS PUTOP3 ;ERROR
1610  RTS
1620 PUTOPN LDY #$00
1630 PUTOP1 LDA ($BB),Y ;GET NAME BYTE
1640  STA $0200,Y ;STORE IT
1650  INY
1660  CPY $B7 ;END OF NAME?
1670  BNE PUTOP1 ;NOT YET
1680  LDX #$00
1690 PUTOP2 LDA PUTSW,X ;GET BYTE
1700  STA $0200,Y ;STORE IT
1710  INX
1720  INY
1730  CPX #$04 ;DONE?
1740  BNE PUTOP2 ;NOT YET
1750  LDA #$61
1760  STA $B9
1770 PUTOP4 STY $B7 ;FILENAME LENGTH
```

Device not present – if no disk drive is connected.
Too many files – if 10 files are already open.
Disk errors – at the end, the disk error channel is read and displayed.
**Use:** PUT is used in conjunction with GET to allow the editing of Commodore assembler source files. PUT can also be used as an alternative save method for Basic programs so that they may be run by using the EXEC command. See EXEC for an example of use.
**Routine entry point:** $997A
**Routine operation:** The filename is read along with the device number and checks are made for missing filename and illegal device number. If these are OK, the file is then opened and each line is output using the print tokens routine to the file. At the end of each line, a carriage return is sent and an extra carriage return inserted at the end of the file. The file is then closed and the disk error channel is read and displayed.

## TYPE

**Abbreviated entry:** T(shift)Y
**Affected Basic abbreviations:** None
**token:** Hex $EE,$1B, Decimal 238,27
**Modes:** Direct and program
**Recommended mode:** Direct
**Purpose:** To display a text file stored on disk to the screen
**Syntax:** TYPE filename,d – where d is the device number (disk only)
**Errors:** Illegal device – if the device number specified is less than eight.
Missing filename – if a null filename is specified.
File not found – if the file does not exist.
Device not present – if no disk drive is connected.
Too many files – if 10 files are already open.
Disk errors – at the end, the disk error channel is read and displayed.
**Use:** TYPE can be used to look at sequential files stored on disk. This can be used rather than GET if you wish to check a certain line in the file is as the file is not loaded in but directly displayed from the disk. Easyscript text files could be just as easily displayed using this routine.
**Routine entry:** $9F50
**Routine operation:** The filename is read along with the

```
1780  LDA #$00
1790  STA $0200,Y
1800  LDY #$02
1810  STA $BB ;POINTER LO
1820  STY $BC ;POINTER HI
1830  LDA #$0E
1840  JSR GETN1 ;GET FILE NUMBER
1850  STA $B8 ;FILE#
1860  STA EXECNO ;FOR EXEC
1870  JSR $FFC0 ;OPEN
1880  BCS PUTOP3 ;ERROR
1890  RTS
1900 PUTOP3 JMP $E0F9 ;OUTPUT ERROR
1910 PUTSW .BYT ',S,W'
1920 ;
1930 PUTCLS LDX #$03
1940  JSR $FFC9 ;OUTPUT TO SCREEN
1950  LDX #$00
1960  JSR $FFC6 ;INPUT FROM KEYBOARD
1970  LDA EXECNO
1980  JMP $FFC3 ;CLOSE FILE
1990 ;
2000 PUTMES LDA #<PMESSG ;
POINTER TO MESSAGE
2010  LDY #>PMESSG
2020  JSR $AB1E ;PRINT MESSAGE
2030  JMP $F5C1 ;PRINT FILENAME
2040 PMESSG .BYT 'WRITING ',$00
2050 .END
```

## TYPE

```
1000 TYPE JSR DPARS ;GET FILE DETAILS
1010  JSR GETOPN ;OPEN FILE
1020  JSR GETIN ;SET INPUT
1030 TYPE2 JSR $FFCF ;INPUT BYTE
1040  LDX $90 ;GET STATUS
1050  JSR $FFD2 ;PRINT BYTE
1060  JSR $FFE1 ;STOP KEY?
1070  BEQ TYPE1 ;YES
1080  TXA
1090  BEQ TYPE2 ;NO ERROR
1100 TYPE1 JMP PUTEND ;DONE
1110 .END
```

device number and checks are made for missing filename and illegal device number. If these are OK, the file is then opened and each character is read in and displayed until the end of file or the stop key is pressed. At this point, the file is closed, the disk error channel is read and the routine exits.

Because of the size of this article it was impossible to print the Basic Loader for the machine code. If you would like a copy of the loader then please send a large stamped self addressed envelope to Better Basic Loader, Your Commodore, 1 Golden Square, London W1R 3AB.

# PROGRAMMING PROJECTS

Garry Marshall explains how a computer can be used to interpret images.

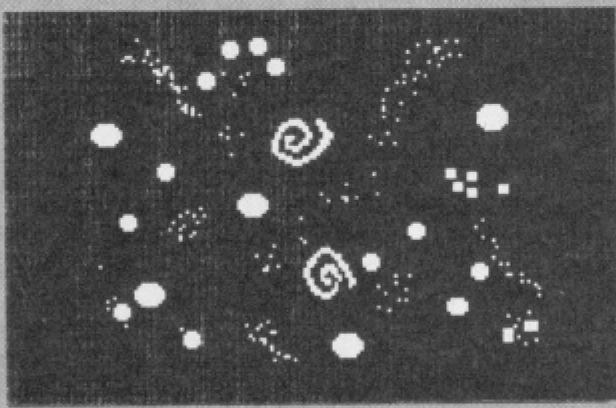


Figure 1 Chaos in the skies

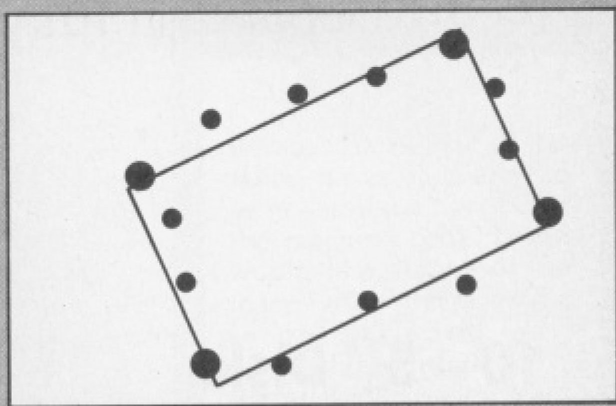


Figure 3 Post-holes with rectangular plan superimposed

IN MANY AREAS, COMPUTERS are used to help interpret images such as that in Figure 1. The illustration shows a part of the sky as seen from a powerful telescope, and computers are used to process pictures of this kind to make sense of them in terms of the galaxies and clusters of objects appearing in them. A similar process takes place in giving a robot the capability to see. To be able to recognise the items that it is to assemble, for instance, an industrial robot must be able to interpret the pattern of darker and lighter dots representing its field of vision as presented by a video camera. In both cases, the computer is running a program that enables it to bring some kind of order to an apparently chaotic scene.

There is another area where computers can be used to help interpret images, and this is in archaeology. Often on a site excavated by archaeologists, the only evidence remaining of a building that once occupied the site is a pattern of holes in the ground. After the building itself has decayed, the holes in which its supporting timbers were placed (known as post-holes) remain. Even for a single building, which will obviously have a rectangular plan, the plan itself is often none too clear. When many buildings have occupied a site at different times with some built over the same ground as earlier, vanished, ones, the overlapping patterns of holes can be chaotic. Computers can be used to good effect in trying to reconstruct the plans of the houses that once occupied a site.

This month's project is to reconstruct the plan of a building from an irregular but more-or-less rectangular pattern of post-holes such as that in Figure 2 by superimposing an outline of the plan on it, as shown in Figure 3.

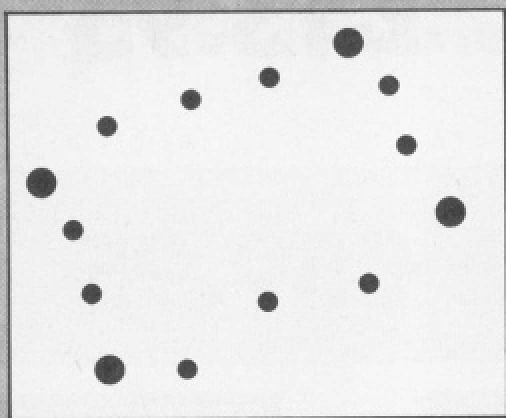


Figure 2 A pattern of post-holes

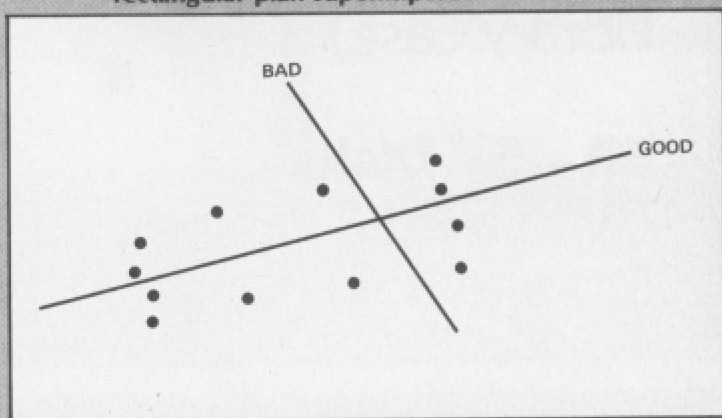



Figure 4: Good and bad approximation to the main axis for a pattern of post-holes.

## The Solution

The first thing the program must do is to plot the pattern of post-holes as in Figure 2. This can be done by reading the positions of the post-holes from DATA statements and then plotting a block at each position to represent a post-hole. The positions will be needed again later in finding the plan of the house, so it is worth storing them in arrays as they are read. Using arrays named XP and YP to hold, respectively, the column and row positions on the screen for the post-holes, and making use of our point-plotting subroutine, which begins with line 1000, the program starts as:

```
10 DIM XP(12), YP(12), X(12), Y(12): NP=12
20 GOSUB 500: REM PREPARE HI-RES SCREEN
30 FOR J=1 TO 12
40 READ C, R: XP(J)=C: YP(J)=R
50 GOSUB 1000: C=C+1: GOSUB 1000: REM PLOT POINT
60 R=R+1: GOSUB 1000: C=C-1: GOSUB 1000
70 NEXT J
80 DATA 46, 55, 62, 62, 99, 77, 120, 90, 109, 105, 106, 118
90 DATA 109, 135, 82, 121, 49, 113, 35, 110, 38, 93, 43, 79
```

The next step is to find the main axis of the house, by doing this we shall find the directions of all the walls of the house. The longer walls will be parallel to the axis and the shorter walls must be at right angles to it. Figure 4 shows the idea behind the method for finding the main axis. It shows a line that goes close to the positions of all the post-holes and a second line that obviously goes much farther from most of them. The first represents a good approximation to the main axis, and the second a poor one. We shall use a well-known mathematical method for finding the line passing closest to a set of points, and this will give us the main axis of the house. The method is that of finding the line giving the 'best least squares fit' to a set of points.

We will write the program so that it waits until a key is pressed before going on to calculate the position of the main axis and to display it using our line-drawing subroutine that starts at line 2000. This gives the next section of the program as:

```
100 GET C$: IF C$=" " THEN
110 FOR J=1 TO 12: X(J)=XP(J): Y(J)=YP(J): NEXT J
120 GOSUB 3000: REM FIND AND DRAW LINE
```

The subroutine that does all the work is:

```
2999 REM FIND AND DRAW LINE
3000 SX=0: SY=0: XX=0: XY=0
3010 FOR K=1 TO NP
```

## Program: Post-Holes

```
10 DIM XP(12),YP(12),X(12),
   Y(12),E(12),C(12): NP=12
20 GOSUB 500: REM PREPARE
   HI-RES SCREEN
30 FOR J=1 TO 12
40 READ C, R: XP(J)=C
   : YP(J)=R
50 GOSUB 1000: C=C+1
   : GOSUB 1000: REM PLOT
   POINT
60 R=R+1: GOSUB 1000: C=C-1
   : GOSUB 1000
70 NEXT J
80 DATA 46,55,62,62,99,77,
   120,90,109,105,106,118
90 DATA 109,135,82,121,49,
   113,35,110,38,93,43,79
100 GET C$: IF C$="" THEN GO
   TO 100
110 FOR J=1 TO 12: X(J)=XP(J)
   : Y(J)=YP(J): NEXT J
120 GOSUB 3000: REM FIND AND
   DRAW LINE
130 GET C$: IF C$="" THEN GO
   TO 130
140 REM FIND DISTANCE FROM
   AXIS
150 MAX=0: MIN=0
160 FOR K=1 TO 12
170 E(K)=YP(K)-M*XP(K)-CC
180 IF E(K)>MAX THEN MAX=E(K)
190 IF E(K)<MIN THEN MIN=E(K)
200 NEXT K
210 REM CLASSIFY EACH POST-H
   OLE
220 FOR K=1 TO 12
230 IF E(K)>0.7*MAX THEN C(K
   )=1: GOTO 300
240 IF E(K)<0.7*MIN THEN C(K
   )=2: GOTO 300
250 IF ABS(M)>1 THEN GOTO 280
260 IF XP(K)>SX/12 THEN C(K)
   =3: GOTO 300
270 IF XP(K)<SX/12 THEN C(K)
   =4: GOTO 300
280 IF YP(K)>SY/12 THEN C(K)
   =3
290 IF YP(K)<SY/12 THEN C(K)
   =4
300 NEXT K
310 FOR Q=1 TO 4
320 NP=0
330 FOR J=1 TO 12
340 IF C(J)=Q THEN NP=NP+1
   : X(NP)=XP(J): Y(NP)=YP(J)
350 NEXT J
360 GOSUB 3000
370 NEXT Q
380 END
500 POKE 53272, PEEK(53272)
   OR 8
510 POKE 53265, PEEK(53265)
   OR 32
520 FOR I=8192 TO 16192
   : POKE I, 0: NEXT I
530 FOR I=1024 TO 2023
   : POKE I, 22: NEXT I
540 RETURN
1000 RO=INT(R/8): CO=INT(C/8)
1010 L=R AND 7
1020 BIT=7 - (C AND 7)
1030 BYTE=8192+RO*320+CO*8+L
1040 POKE BYTE, PEEK(BYTE) O
   R 2^BIT
1050 RETURN
2000 DX=X2-X1: DY=Y2-Y1
2010 IF DX=0 THEN 2070
2020 FOR C=X1 TO X2 STEP SGN
   (DX)
2030 R=INT(Y1+(C-X1)*DY/DX)
2040 GOSUB 1000
2050 NEXT C
2060 RETURN
2070 C=X1
2080 FOR R=Y1 TO Y2 STEP SGN
   (DY)
2090 GOSUB 1000
2100 NEXT R
2110 RETURN
2999 REM FIND AND DRAW LINE
3000 SX=0:SY=0:XX=0:XY=0
3010 FOR K=1 TO NP
3020 SX=SX+X(K): SY=SY+Y(K)
3030 XX=XX+X(K)*X(K)
   :XY=XY+X(K)*Y(K)
3040 NEXT K
3050 D=SX*SX-NP*XX
3060 M=(SY*SX-NP*XY)/D
3070 CC=(SX*XY-SY*XX)/D
3080 X1=200: X2=0
3090 FOR K=1 TO NP
3100 IF X(K)<X1 THEN X1=X(K)
3110 IF X(K)>X2 THEN X2=X(K)
3120 NEXT K
3130 Y1=M*X1+CC: Y2=M*X2+CC
3140 GOSUB 2000: REM DRAW
   LINE (X1,Y1) TO(X2,Y2)
3150 RETURN
```

```
3020 SX=SX+X(K): SY=SY+Y(K)
3030 XX=XX+X(K)*X(K): XY=XY+X(K)*Y(K)
3040 NEXT K
3050 D=SX*SX-NP*XX
3060 M=(SY*SX-NP*XY)/D
3070 CC=(SX*XY-SY*XX)/D
3080 X1=200: X2=0
3090 FOR K=1 TO NP
3100 IF X(K) < X1 THEN X1=X(K)
3110 IF X(K) > X2 THEN X2=X(K)
3120 NEXT K
3130 Y1=M*X1+CC: Y2=M*X2+CC
3140 GOSUB 2000: REM DRAW LINE (X1, Y1) TO (X2, Y2)
3150 RETURN
```

With the main axis of the house plan located, we can identify the post-holes that belong to the long walls because those on one long wall will all be more or less the same distance above it and those on the other about the same distance below it. The remaining post-holes must belong to the short walls and, broadly speaking, those to the right of the centre of the building will be on one short wall and those to the left on the other.

By introducing a code for the walls of the plan, we can use an array named C to hold a code number for each point and so record for each point the wall to which it belongs. The code is:

| Wall | Number |
|---|---|
| Long wall | 1 |
| Long wall 2 | 2 |
| Short wall 1 | 3 |
| Short wall 2 | 4 |

Using this code, the assignment C(1)+4 would mean that post-hole 1 is on short wall 2.

The next program segment, also written so as to be initiated by pressing any key, is:

```
130 GET C$: IFC$=" " THEN GOTO 130
140 REM FIND DISTANCES FROM AXIS
150 MAX=0: MIN=0
160 FOR K=1 TO 12
170 E(K)=YP(K)-M*XP(K)-CC
180 IF E(K) > MAX THEN MAX=E(K)
190 IF E(K) < MIN THEN MIN=E(K)
200 NEXT K
210 REM CLASSIFY EACH POST-HOLE
220 FOR K=1 TO 12
230 IF E(K) > 0.7*MAX THEN C(K)=1: GOTO 300
240 IF E(K) < 0.7*MIN THEN C(K)=2: GOTO 300
250 IF ABS(M) > 1 THEN GOTO 280
260 IF XP(K) > SX/12 THEN C(K)=3: GOTO 300
270 IF XP(K) < SX/12 THEN C(K)=4: GOTO 300
280 IF YP(K) > SY/12 THEN C(K)=3
290 IF YP(K) < SY/12 THEN C(K)=4
300 NEXT K
```

After this, not forgetting to declare the arrays E and C, it only remains to draw the lines for each side of the building. This can be done by using the subroutine for finding and drawing a line four times, once each time for the points on each wall. This is done by the final section of the program, which is:

```
310 FOR Q=1 TO 4
320 NP=0
330 FOR J=1 TO 12
340 IF C(J)=Q THEN NP=NP+1: X(NP)=XP(J): Y(NP)=YP(J)
350 NEXT J
360 GOSUB 3000
370 NEXT Q
```

The complete program for drawing the plan of a house from an approximately rectangular pattern of post-holes is listed in Figure 5. Remember that when it is run it draws the pattern of post-holes and then waits for you to press a key before it draws the main axis. Then it waits again for you to press a key before it draws the walls.

## Further Developments

The way to investigate the capabilities of this program is to try it out with different sets of data. Trying several sets will give a feel for what is meant by phrases such as 'more or less rectangular' and 'more or less the same distance away'. If you try changing the number 0.7 in lines 230 and 240 at the same time, you will find that you can adapt the program to deal successfully with any reasonable data. The next step, of course, is to generalise the program so that it adapts itself to the data it is given. I think you will find that this program shows a degree of 'intelligence', and if you can make it adapt itself to different situations, you will have made it more intelligent.

**Runecaster delves into some dungeons and dragons games and faces danger and death.**




## New and Devious...

GLOBAL IS A RELATIVELY NEW NAME in computer software and if the company's first adventure game – *The Magician's Ball* – is anything to go by it will be worth watching out for in the future. It is by no means perfect as there are several anomalies apparent in playing it... but nevertheless the presentation and the command structure are good and sufficiently different to make it worth your attention.

The program is on cassette only and will run on the C64 (or C128). A fast loader is incorporated and the program will load in just over four minutes. The screen display is colourful and the text easily read, with the graphic picture of your location occupying a small rectangle in the top left corner.

The graphics are not particularly exciting but are clear and are 'drawn' very rapidly. Occasional use of sprites adds some movement that gives a little life to the scenes. Unlike some games the pictures, although simple are quite different and easily give quick visual recognition to one's location.

The main location description is to the right of the picture and contains plenty of information to build quite a fair mental image of your surroundings. Visible exits are indicated and input commands are entered at the bottom of the screen.

The 12 line 'window' between the description and the command line will contain additional information on what is seen at that location – creatures, objects that can be carried and even what other creatures are carrying!

Briefly, the storlyine is that an evil magician has abducted a beautiful princess from her father's side; you – Caro – were there at the time and in a foolish attempt to rescue the girl...got swept away by the magician's magic as he returned to his own domains. You are now in the magician's realm and...on your own!

Input commands may consist of reasonably complex sentences and also permit you to tell other characters what to do. A very interesting feature of this adventure is that you can swop personalities and 'control' some of the other characters you meet in your travels.

The normal 'save game' facilities are actioned with 'saving' and 'loading' times of just under two minutes but another neat trick from Global is the 'Quicksave' and 'Quickload'. These are implemented by pressing the Commodore key and either 'S' or 'L'.




This 'save' option almost instantaneously stores your present position into a protected area of memory. A very useful command if you think you are about to be killed or even to create a 'fall back' position if you are not sure of your next few moves.

Further 'user friendly' options exist in the input command area – not only can you delete an unwanted letter by using the delete key in the normal manner but you can delete the entire command with SHIFT and CLR.

Pressing '@' repeats the last command and keying the left arrow key (top left of the keyboard) recalls the last command for you to modify if desired. I think it is facilities such as these that make Global worth watching in the future. User friendly games are worth cultivating!

The operating system may be user friendly, the game is in many ways just plain devious! All the clues are there but you may have difficulty seeing the wood for the trees. There are several 'red-herring paths' and the number of objects you can carry is not always as great as you may wish – perhaps you should get someone/something to carry them for you!

Throughout your journey, you travel to musical accompaniment from 'Tubular Bells', this provides a pleasant interlude as you sit back to ponder your next move.

There are the anomalies I mentioned earlier but these do not alter the game play in any way – they mainly appear as various characters saying something that has no useful or significant purpose!

There are also a number of situations where you will have at one location, two (or more) items that although different are similar – a gold and an ornamental key for instance. It is worth dropping the one you do not want to use at another location as the program sometimes finds it difficult to understand what you want it to do!

*The Magician's Ball* is good fun and both novice and experienced adventurers should find something in it to enjoy.

## Old But Faster

Once upon a time there were no C64s or C128s. I know that is difficult to believe but it's true. There were PETs and Ataris and TSR 80s and quite a number of programs for them.

Amongst these there were also a number of adventure games that closely followed the general idea of Dungeons and Dragons, with lots of monsters to meet in battle and treasure to be found by the bold and daring.

They caught the imagination of thousands of home computer users in America (the land of their origin) and there were even competitions and conventions where adventurers of like mind could compete against the clock to prove their prowess in the worlds of dungeons and magical tombs.

One of the favourites was called *The*

*Temple of Apshai*. At that time, with the exception of the Atari, none of the home computers had what we could call hi-res graphics and as it was not written for the Commodore machines anyway, the conversion for the PET was sadly lacking.

The graphics showing the rooms and corridors around your character, took an age to be 'drawn' to the screen, the action was slow, the response to your commands (and there were plenty of options) was painfully slow and last but not least the keyboard buffer was not cleared.

This last 'fault' was diabolical, because, with the slow response to the input commands (all single key) there was a high probability that you would press one or more keys several times trying to either evade or attack a monster with the consequence that your character would still be striking at the monster long after it had been slain!

Even a fairly recent conversion for the C64 still exhibited many of these shortcomings making the game tedious and boring to play. All these half-hearted attempts were really a great shame because the original concept was worth playing.

I am pleased to report that at long last, a playable version has appeared for the C64. This is a new (yet another!) interpretation entirely, with far superior graphics, faster playing all round, no keyboard buffer problems and to cap it all, not only *The Temple of Apshai* but also on the same disc, the two additional adventures that followed: *Upper Reaches of Apshai* and the *Curse of Ra'*.

The *Temple of Apshai Trilogy* is produced by Epyx Computer Software (*Impossible Mission, Summer Games* etc.) and is distributed in the UK by US Gold. One other vital fact is that the three-on-one-disk will retail for only £9.95!

If you have the slightest inclination towards 'monster bashing', then these programs are for you. Even if you are only barely interested then the Temple Trilogy is worth considering, purely as being a classic of its type.

The instruction manual for the trilogy is some 80 pages long and covers all the basic rules for playing, together with a room by room description of the various locations. The graphics representation shown on the screen simply depicts a series of rooms and joining corridors.

It is not absolutely necessary to read the descriptions but it helps set the scene. You will also need to refer to the manual to cross-check against the various treasures that you find.

The three scenarios provided by the Temple Trilogy are in many respects compatible with many non-computer role playing games. You may even 'convert' an existing character in one of these to venture forth into Apshai. When you first start playing you are given several choices.

You may start from scratch, with your characters given a random rating (from 1-18) for their six basic attributes: intelligence, intuition, ego, strength, constitution and dexterity, together with a limited amount of silver pieces with which to equip themselves.

You may enter a previously 'saved' character from disk or you may enter the attributes of a known character from some other game. Or you may restore a previous game 'saved' at any point within Apshai.

Your character is 'named' and is your own 'persona' within the confines of the game. If you start from scratch then you will have to buy the various pieces of equipment you think you will need – from the innkeeper.

This is an art in itself as you must learn to haggle with him to get the best bargains. Pay the asking price and you will end up with very little equipment!

Choice of equipment is fairly important. Various swords are on offer but if you do not have a great deal of strength then a two handed sword will tire you quickly. A small shield may be very effective if you have high 'dexterity'.

The status screen shown throughout your adventures in Apshai shows wounds and fatigue, the lower percentage of wounds the nearer you are to death. The lower percentage of fatigue the less energy you have to fight or run away.

Wounds may be healed over the passage of time or through the use of 'healing salves' or 'elixirs', the former may be purchased from the innkeeper, the latter can be found within the rooms and corridors of Apshai.

Fatigue may be restored by moving very slowly, resting or by firing arrows! The heavier your equipment or treasure, then the more fatigued you will become.

Having beaten the innkeeper down in price and purchased a sword, armour, a shield, bow, arrows and a few healing salves, you are ready to enter Apshai. You have the choice of which realm to explore and at what level.

Each realm has four levels and for Dungeons and Dragons followers levels one to two are appropriate to third level characters and the lower levels are more suited to fourth and sixth level characters.

Once you have set out on your present quest you have a number of commands available to you. The latest version of the Trilogy allows for the use of a joystick, with the four directional movements giving you 'turn 90 degrees left or right', move five feet in the direction you are facing or rest.

The joystick, with the fire button depressed, also offers normal attack, parry, thrust (all out attack) or fire a normal arrow.

Single key commands from the keyboard also allow you to: move forward from one to nine feet (keys 1-9), turn to the left, right or turn about (180 degrees), attack, thrust or parry, fire a normal or a magic arrow.

There are also a number of special commands you may use, such as examining a wall for secret doors, opening doors, searching for traps, picking up treasures, listening for monsters, talking to monsters, healing yourself with salves or elixirs, checking the inventory of your treasures or saving the game or character for future use.

As you may now be beginning to understand, the *Temple of Apshai Trilogy* is a little more than a straight-forward 'monster bashing' adventure game. A little thought is also required if you are to last any length of time within its dangerous environs.

The option of defining your own character has got to be double edged. It is all too easy to give your character 18 points for all attributes. This strikes too close to, dare one say...cheating?

On the other hand, pressing RUN/STOP RESTORE if the attributes given to you by the innkeeper are particularly poor does give you the chance to improve on the random attribute system...after all there are various magical items to be found that will increase some if not all of your vital statistics!

Most levels of each realm have between 50 and 60 different rooms, so there is plenty to explore. Some monsters are 'tied' to particular locations but there are also plenty that wander around just looking for the odd human.

If you choose the right realm and level it is not too difficult to increase your experience and treasure but venture deeper and things become a little more difficult!

Should the worst happen, then there is still a chance that all is not lost. If you come to an untimely end one of four things may happen, you may get eaten by a wandering monster... bye-bye, or you may be found and resurrected by one of three 'friends of the dungeon'.

One will do this for no cost, other than the request of a donation to his holy order, the other two will extract varying degrees of payment. So what? You are still alive and able to add to your valuable experience.

If you 'save' your character after each foray into the unknown, you should be able to gradually build up a better powerful character. After each return to the inn, the innkeeper will pay you for any treasure you have collected and you will have the chance of adding or improving your equipment.

Just remember the restrictions that super heavy weapons and armour place upon your movement and energy usage.

Also remember that copies of the poorer version of *Temple of Apsahi* are still around, make sure you get the latest version. You want to spend weeks over the game not hours over one move! Happy hunting!

# COMMUNICATION CORNER

**News, views, chat and all the latest services from David Janda.**

ITS BEEN A HECTIC MONTH IN THE world of Comms. Some services have decided to put up their prices, new services have been announced and new modems are on the horizon. The fact that I managed to tip half of my dinner over my 64 thus putting it well out of service didn't help things either!

## Chatting!

First off, Micronet's new Mainframe Chatline went into operation on the 1 November. After a shakey start with everyone complaining left, right and centre things have been put in order.

As well as the new Chatline service (called Quickchat) the 'old' style Chatline (now called Daisychat) has been spruced up a bit. The setup is as follows.

There are currently six new Quickchat 'channels' in operation. Operated on the new Prestel computer, each message is displayed almost instantaneously after it has been sent. Once you bash in a message (colour and graphics allowed) and send it, keying #2 will display the Quickchat frame. As messages are sent they are displayed and overwrite the previous message. At present there is no charge for using the service, but I understand that non-Micronet subscribers do pay a few pence per message.

The channels covered are; one and two general, three politics, four religion, five gay line and six music line.

Those of you who preferred the old style of Chatline where you could read previous messages can use the Daisychat service of which there are two channels. There is a new layout for the message frame and the service still costs 2p a message. The really good news is that it's much faster in updating messages.

## Buttons

Buttons 700, the new quiz area on the Net has also started. It has open access to all Prestel subscribers and offers a number of quizzes and games to play. This area will no doubt make money for the Net as there is many a travel agent's assistant who likes spending his bosses hard earned cash playing Prestel quizzes. All games/quizzes are half price to Micronet subscribers.

Buttons also has a Chatline of its own. The interesting point about it is that it's open access to all Prestel subscribers, and I must say they the level of conversation is...er, interesting! A bit of a rip-off at 7p a message though!

## Compunet Costs

Did you know that Compunet has had its new three tier subscription rate in operation since 1 September? No? Well it would appear that a lot of people haven't so here are some (brief) details.

The basic rate subscription costs £7.50 per quarter and provides you with free courier service and six hours free connect time per quarter. Unless you start a cash account where you send a cheque to Compunet of X amount you will not be able to do any uploads or purchase items on the system. This is the subscription level given free with Commodore modems.

For those who are not yet subscribers, take it from me that this type of subscription is duff, you are far better off with the 'standard rate' subscription. This provides free courier service, free unlimited off-peak connection and 500 free uploaded frames. Basic rate subscribers can 'upgrade' to standard rate (which costs £10 per quarter) and will be charged £2.50 per quarter.

With a Standard rate subscription it is possible to set up a credit account in which you pay £50 by standing order to Compunet. You can then upload and download to your heart's content until your credit runs out.

Confused? I am, so why not call Compunet on its new number which is 01-965-8866.

More on CNET proper next month when I get my 64 fixed!

## The News!

By the time you read this, Modem House may have announced a major deal with CNET. It is rumoured that Mod House has designed a chip that fits inside the OEL comms package. This would break the monopoly that Commodore has on modems accessing CNET – details next month.

Meedmore Distribution Ltd is developing a modemless modem for around 25 quid. Available next February, the unit consists of a disk and cable which contain the bits of electronics. Apparently you plug one end into the cassette port of the C64 and the other into the BT socket on the wall, and voila! 300/300 baud full duplex modem!

Meanwhile, the big boys, viz Miracle Technology, have produced the C64 Multi Modem for £113. Featuring 1200/75 and 300/300 baud with auto-answer and auto-dial, it connects to the cartridge port on the C64. Software included (on ROM) gives dumb terminal and Prestel compatibility.

## The Wind Up

Have I not informed you of something? Do you know something that is happening in the Comms world that I don't? Then inform me (please!) by sending a message to me on Prestel 919992677 or Compunet ID D.JANDA. Alternatively, you can send me a letter (a what?) addressed to moi c/o *Your Commodore*.

# · AIR · COMBAT · EMULATOR ·

· BY CASCADE GAMES ·

1-3 Haywra Crescent Harrogate

**£10.95**
**INC. FREE WATCH AND POSTAGE**

# THE FASTEST GAME ON TWO WINGS! AND A FREE MULTIFUNCTION CALCULATOR WATCH




The AIR COMBAT EMULATOR, packed with exciting features, is by far the fastest, most realistic flight simulator available. For the first time on a home computer, feel what it is **really** like to fly a high performance military jet and engage tanks, missiles, ships, helicopter gun-ships and two types of cunning intelligent enemy aircraft.

Featuring over 300 sprite images, ACE has the smoothest, most detailed cockpit view of hills, trees and other ground objects ever achieved thanks to specially developed Dynamic Graphic Technique (D.G.T.©) allowing sensational high speed **solid** 3D graphics.

In combat, your head will spin as you struggle with the help of the advanced avionics which include a rear-view camera and radar, to stay alive long enough to do some damage to the enemy.

The unique two seat flying option (pilot, navigator/weapons man) and Synchro Speech© which passes vital aircraft condition information to the flight crew without freezing the action combined with dynamic tactical map, take offs and landings, summer, winter and night-time flying and a remarkably realistic in-flight refuelling option makes ACE the ultimate AIR COMBAT EMULATOR.

### *ACE SCENARIO*

A vast invasion fleet has landed on your southern shores and discharged scores of tanks supported by helicopter gun-ships and protected by mobile surface to air missiles. Squadrons of enemy fighters give the enemy forces air cover. Advancing ruthlessly they are conquering your homeland, facing no opposition.

You are the last fighter pilot with only three A.C.E. Mark 2.1 multi role All Weather All Terrain (AWAT©) combat aircraft. Based at an allied airbase, your task is to attack and destroy the numerically superior enemy land and air forces and drive them from your shores. The final stage of the conflict, having already destroyed the enemy land forces and shot down their air force, is to sink the enemy fleet as it evacuates its defeated army.

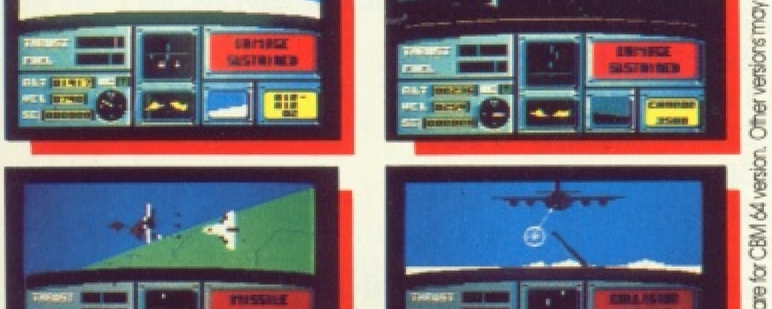
### *ACE – EXPERIENCE IT NOW*

Every computer game fan will love ACE – just look at these features:-

- 100% optimized machine code uses all available memory.
- Featuring Dynamic Graphic Techniques (D.G.T.©)
- The smoothest, fastest and most detailed cockpit view of any flight simulation.
- Views of hills, trees, tanks and other ground objects. Aerial combat, ground attack and naval attack.
- Synchro Speech© ie. the speech does not freeze the action.
- Two seat option – fly and fight with the aid of a friend.
- Sensational 3D solid (not boring wireframed) graphics.
- High score saving system.
- AWAT© operation (All Weather All Terrain).

* Note due to memory limitations ACE on the C16 and VIC20 (+8K Ram) do not have any ground objects.

**FREE 20 KEY MULTIFUNCTION CALCULATOR WATCH**
**• 12/24 Hour Time Mode • Alarm Mode • Calendar Mode • Calculator Mode**




Screen shots are for CBM 64 version. Other versions may vary.

## *ACE – EXPERIENCE IT NOW!*

# FREE!!!

**Order ACE today and receive this fantastic 20 KEY MULTIFUNCTION CALCULATOR WATCH**

**ABSOLUTELY FREE!!!**

PRICE **£10.95 (CBM 16 and VIC 20 £9.95)**

Postage FREE in U.K. Add £1 for overseas orders.
Allow 28 days for delivery.
I enclose cheque/postal order for (£..............................) made payable to Cascade Games Ltd.
Or through any Post Office by Transcash (Giro No. 655 6655).
For even faster ordering when charging to Access, Barclaycard and Trustcard Visa use our 24 hr service (Tel. 0423 504663).

ENTER CREDIT CARD NO. IN BOX:-

VISA

☐ C64 ☐ C16 ☐ C VIC 20 (with 8K RAM expansion)
☐ C PLUS 4 ☐ C128

Please tick box

NAME ____________________

ADDRESS ____________________

POSTCODE ____________________

COUNTRY ____________________



Please complete and send to:-
**Cascade Games Ltd.**
1-3 Haywra Crescent
Harrogate HG1 5BG. England.
Tel. (0423) 504663
Registered No. 1755554

cascade

YC 2

# C16 KEY



**Derek Moody gives more control to your fingers with this article for the C16.**

I PURCHASED A C16 AT THE end of March but horror of horrors, here was yet another Micro supplied with an inadequate manual. Commodore suggest that you should buy their Programmer's reference guide, but that doesn't even contain a memory map, let alone operating system entry points or connector pinouts. First of all, I needed a decent keyboard control routine, so I disassembled the ROM and starting searching for the necessary information. This article is based on some of the results.

The C16 detects keypresses and stores them, even when the computer is doing something else. This is possible because the keyboard is being read in an interrupt routine, this routine also maintains the real time clock and does a certain amount of housekeeping for the operating system. The IRQ interrupt routine is called 60 times each second, and there are three points at which it is vectored through RAM, at $312, $314, and $316. The vector that will be of most interest to us is at $312, the computer refers to it after most of the housekeeping, but before updating the real time clock and reading the keyboard. A vector, by the way, is an address held in two bytes of RAM, which points to a block of machine code in ROM, by altering a vector, the programmer can cause his own block of code to be used instead.

The keyboard is read by a short, 10 byte routine at $DB70 this works by writing the contents of the accumulator to the columns of the keyboard matrix, and reading the rows

**Program Listing 1**

```
10 REM C16 KEYBOARD MATRIX DEMO
11 REM
12 REM BY DEREK MOODY APRIL 1984
13 REM
20 :
40 TC%=3072+41
50 D%=14335
60 GOSUB 3000
97 :
98 REM CONTROL ROUTINE
99 :
100 GOSUB 2000
110 DO
120 : GOSUB 1000
130 LOOP
140 END
997 :
998 READ MATRIX, AND PLOT RESULTS
999 :
1000 XX%=1
1010 FOR X=0 TO 7
1020 : POKE D%,XX%
```

## Program Listing 1 (cont.)

```
1030 : SYS (D%+1)
1040 : R%=PEEK(D%)
1050 : YY%=1
1060 : FOR Y=0 TO 7
1070 :    C%=32
1080 :    IF (R% AND YY%) THEN C%=160
1090 :    POKE TC%+Y*120+X*3,C%
1100 :    YY%=YY%+YY%
1110 : NEXT Y
1120 : XX%=XX%+XX%
1130 NEXT X
1140 RETURN
1997 :
1998 PRINT MATRIX FORMAT ON SCREEN
1999 :
2000 SCNCLR
2010 PRINT"I/D 3  5  7  9 ";CHR$(109);CHR$
(110);"  ";CHR$(95);"  1 "
2020 PRINT:PRINT
2030 PRINT"RET W  R  Y  I  P  * C/H"
2040 PRINT:PRINT
2050 PRINT" #  A  D  G  J  L  ; CTR"
2060 PRINT:PRINT
2070 PRINT"HLP 4  6  8  0  ^ ->  2 "
2080 PRINT:PRINT
2090 PRINT"F1  Z  C  B  M  . ESCSPC"
2100 PRINT:PRINT
2110 PRINT"F2  S  F  H  K  :  = COM"
2120 PRINT:PRINT
2130 PRINT"F3  E  T  U  O  -  +  Q "
2140 PRINT:PRINT
2150 PRINT" @ SHF X  V  N  ,  / R/S"
2160 RETURN
2997 :
2998 INITIALISE MACHINE CODE
2999 :
3000 FOR PTR=D%+1 TO D%+14
3010 : READ CODE%
3020 : POKE PTR,CODE%
3040 NEXT PTR
3050 RETURN
3497 :
3498 MACHINE CODE DATA
3499 :
3500 DATA 173 , 255 , 55 , 73 , 255 , 32 ,
112 , 219 , 73 , 255 , 141 , 255 , 55  , 96
3997 :
3998 DISASSEMBLY OF MACHINE CODE
3999 :
4000 :   3800  AD FF 37 LDA $37FF
4010 :   3803  49 FF    EOR #$FF
4020 :   3805  20 70 DB JSR $DB70
4030 :   3808  49 FF    EOR #$FF
4040 :   380A  8D FF 37 STA $37FF
4050 :   380D  60       RTS
```

back into the accumulator, the X and Y registers are preserved.

The keyboard matrix is shown in Figure 1, to select a column for reading, that column should be pulled low by writing a zero into the appropriate bit, whilst all the other columns should be held high, ie., bit value 1. Therefore to select column 2, the number required is, in binary 11111011 or $FB or decimal 251. If no key on that column has been pressed, then the number returned will be 11111111 or $FF or decimal 255. If, however any key on that column has been depressed, then the appropriate bit(s) will be zero, for example, if both 'C' and 'T' were pressed, then the number would be 10101111 or $AF or decimal 175. To examine every key, requires the routine to be called eight times, once for each column. To detect whether ANY key has been pressed, send a zero to all columns, and if the returned value is not 255 then one, or more, keys must be pressed, although which key(s) will not be obvious.

The advantage of using this routine rather than the BASIC GET and GETKEY statement lies in the ability to detect and use multiple keypresses. PROGRAM 1 serves as a useful example and testbed. When RUN the program presents



## Program Listing 2

```
10 REM C16 GOBBLERS
11 REM
12 REM BY DEREK MOODY APRIL 1985
13 REM
20 :
30 DIM PL%(1,500)
40 TC%=3072
50 SCNCLR
60 D%=14335
70 GOSUB 2500
80 POKE P1%,81
90 POKE P2%,87
97 :
98 REM CONTROL ROUTINE
99 :
100 DO UNTIL T
110 :  GOSUB 200
120 :  GOSUB 600
130 :  GOSUB 300
140 LOOP
150 IF T=P1% THEN S2=S2+100:ELSE S1=S1+100
180 GOSUB 2000
190 END
197 :
198 GET MOVES AND UPDATE POSITIONS
199 :
200 SE%=2
210 GOSUB 500
220 P1%=P1%+MOVE%
230 SE%=32
240 GOSUB 500
250 P2%=P2%+MOVE%
260 RETURN
297 :
298 INSERT RANDOM 'STARS'
299 :
300 IF INT(RND(1)*10) THEN RETURN
310 ST%=TC%+INT(RND(1)*1000)
320 IF PEEK(ST%)<>32 THEN RETURN
330 POKE ST%,42
340 RETURN
497 :
498 CALL KEYBOARD ROUTINE AND
CALCULATE MOVE 499 :
500 POKE D%,SE%
510 SYS (D%+1)
520 R%=PEEK(D%)
530 MOVE%=0
540 IF (R% AND 2) THEN MOVE%=MOVE%-40
550 IF (R% AND 4) THEN MOVE%=MOVE%-1
560 IF (R% AND 16) THEN MOVE%=MOVE%+40
570 IF (R% AND 32) THEN MOVE%=MOVE%+1
580 IF MOVE% THEN SOUND INT(SE%/30+1),200+SE%,4
590 RETURN
597 :
```

**Figure 1 — Keyboard Matrix**

| ROWS \ COLUMNS | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | INST DEL | 3 | 5 | 7 | 9 | ↓ | ← | 1 |
| 1 | RET | W | R | Y | I | P | * | CLR HOME |
| 2 | £ | A | D | G | J | L | ; | CTRL |
| 3 | HELP | 4 | 6 | 8 | Ø | ↑ | → | 2 |
| 4 | F1 | Z | C | B | M | . | ESC | SPACE |
| 5 | F2 | S | F | H | K | : | = | C= |
| 6 | F3 | E | T | U | O | – | + | Q |
| 7 | @ | SHIFT | X | V | N | , | / | RUN STOP |

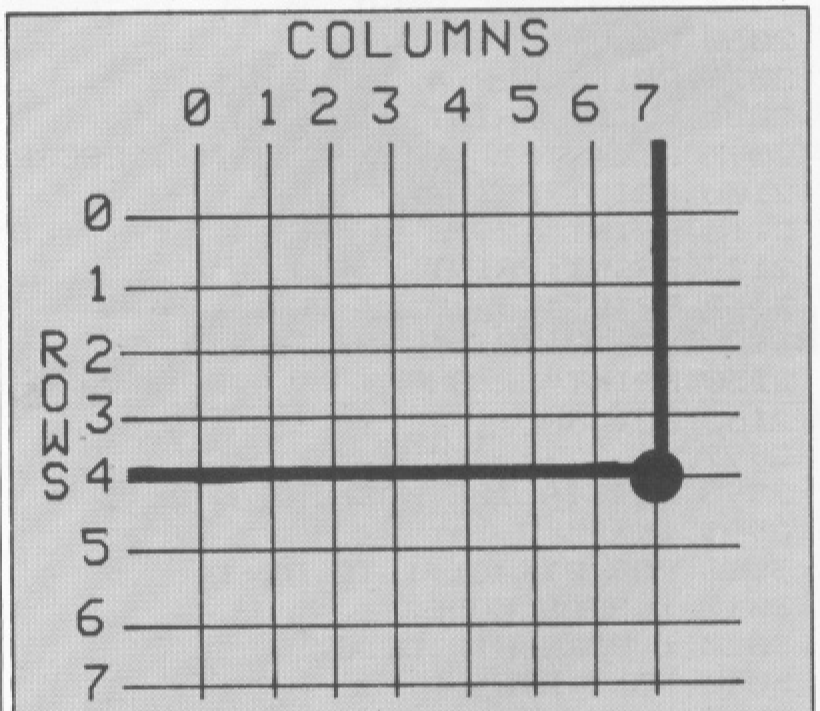
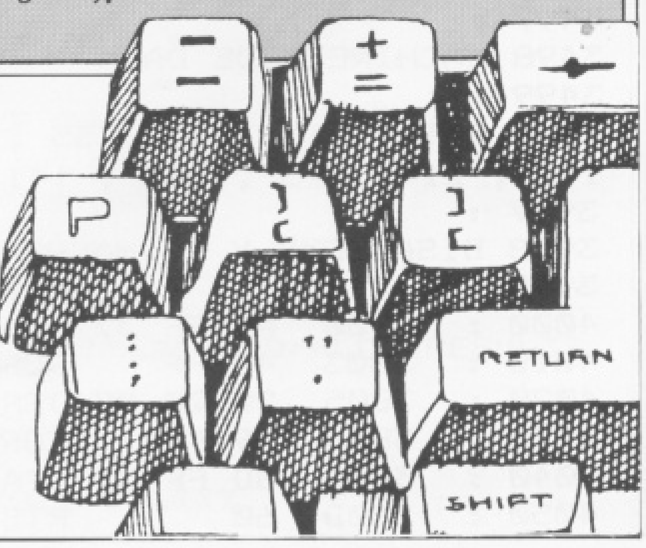



Figure 2 — Single Keypress



## Program Listing 2 (cont.)

```
598 UPDATE SCREEN AND LOOK FOR COLLISIONS
599 :
600 T1%=PEEK(P1%)
610 T2%=PEEK(P2%)
620 IF T1%>44 THEN IF P1%<>PL%(0,S1%) THEN T=P1%:RETURN
630 IF T2%>44 THEN IF MOVE% THEN T=P2%:RETURN
640 GOSUB 700
645 PL%(0,S1%)=P1%
650 PL%(1,S2%)=P2%
660 POKE P1%,81
670 POKE P2%,87
680 POKE PL%(0,E1%),32
690 POKE PL%(1,E2%),32
695 RETURN
697 :
698 HANDLE ARRAY AND TAIL
699 :
700 S1%=S1%+1:IF S1%>500 THEN S1%=0
710 S2%=S2%+1:IF S2%>500 THEN S2%=0
720 IF T1%<>42 THEN E1%=E1%+1:IF E1%>500 THEN E1%=0
730 IF T2%<>42 THEN E2%=E2%+1:IF E2%>500 THEN E2%=0
740 IF T1%=42 THEN SOUND 1,500,10:S1=S1+10
750 IF T2%=42 THEN SOUND 2,300,10:S2=S2+10
760 RETURN
997 :
998 SETUP SCREEN AND INITIALISE PLAYERS
999 :
1000 FOR X=0 TO 39
1010 : POKE TC%+X,102
1020 : POKE TC%+960+X,102
1030 NEXT X
1040 FOR X=39 TO 959 STEP 40
1050 : POKE TC%+X,102
1060 : POKE TC%+X+1,102
1070 NEXT X
1080 P1%=TC%+490
1090 P2%=TC%+510
1100 S1%=1
1110 S2%=1
1120 E1%=0
1130 E2%-
1140 T=0
1150 PL%(0,S1%)=P1%
1160 PL%(1,S2%)=P2%
1170 GOSUB 3000
1180 S1=0
1190 S2=0
1200 VOL7
1210 RETURN
1997 :
1998 PRINT SCORES AND END GAME
1999 :
2000 PUDEF"0"
2010 PRINTCHR$(19)
2020 PRINT
```

an on-screen picture of the keyboard matrix, if any keys are held down, then the relevant matrix position is indicated, note that in this case the keys must be held down as the BASIC routine takes a little over a second to record the current position. As the programme is written, it leaves the operating system keyscan untouched, when you have got the programme running properly, add the following line.

70 POKE 786,190:POKE787,252

SAVE this version of the programme before you RUN it, this steals the vector at $312, and points it to the end of the interrupt routine, thus bypassing the normal keyscan, and leaving BASIC with no way of reacting to the keyboard. The new version of the programme does not recognise the RUN STOP key, and allows us to experiment at will.

In PROGRAM 1, TC% points top the top left corner of the matrix as it is printed on the screen. D%, points to the data byte through which parameters are passed to and from the machine code routine, the machine code itself starts at D%+1. XX% is the value that is passed to the machine code routine, R% is the value returned. YY% is a value generated for comparison with

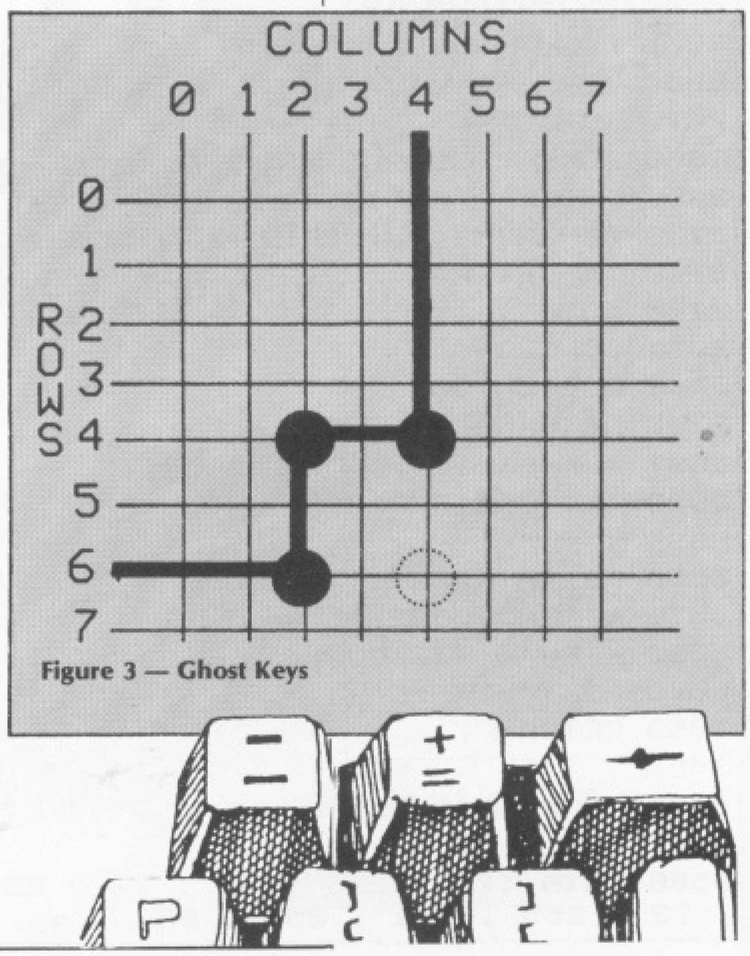
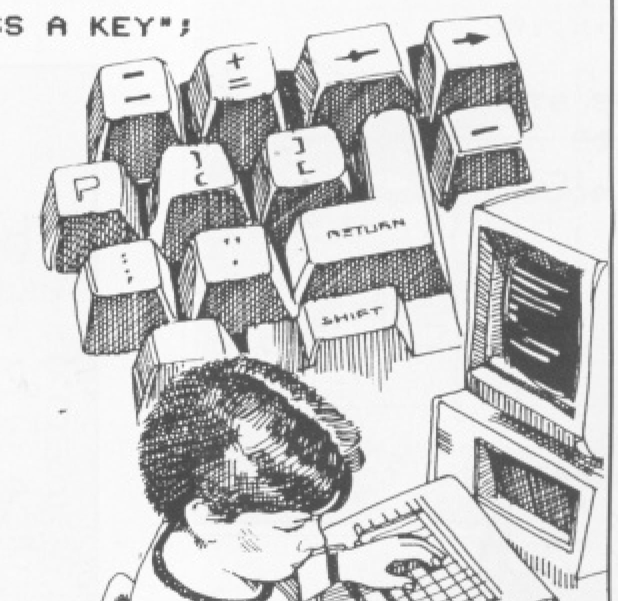



**Figure 3 — Ghost Keys**

**Program Listing 2 (cont.)**

```
2030 PRINTCHR$(166);"          #####################"
2040 PRINTCHR$(166);"          #                   #"
2050 PRINTCHR$(166);"          #    ";
2060 PRINTUSING"####";S1;
2070 PRINT"    ";
2080 PRINTUSING"####";S2;
2090 PRINT"  #"
2100 PRINTCHR$(166);"          #                   #"
2110 PRINTCHR$(166);"          #####################"
2120 FOR X=0 TO 100
2130 :  GET A$
2140 SOUND 3,33,1
2150 NEXT X
2160 RETURN
2497 :
2498 TITLE SCREEN
2499 :
2500 SCNCLR
2510 FOR X=0 TO 15
2520 :  PRINTSPC(X*2);
2530 :  PRINT"GOBBLERS"
2540 NEXT X
2550 PRINT
2560 PRINT"  PLAYER 1                     PLAYER 2"
2570 PRINT
2580 PRINT"      W          -UP-         P"
2590 PRINT"      Z         -DOWN-        ."
2600 PRINT"      A         -LEFT-        L"
2610 PRINT"      S         -RIGHT-       :"
2620 PRINT
2630 PRINT"              PRESS A KEY";
2640 PRINTCHR$(19)
2650 FOR X=0 TO 11
2660 :  PRINT
2670 NEXT X
2680 PRINT" GOBBLE THE STARS"
2690 PRINT"  AVOID THE REST"
2700 DO WHILE A$=""
2710 :  GET A$
2720 LOOP
2730 SCNCLR
2740 GOSUB 1000
2750 RETURN
2997 :
2998 INITIALISE MACHINE CODE
2999 :
3000 FOR PTR=D%+1 TO D%+14
3010 : READ CODE%
3020 : POKE PTR,CODE%
3040 NEXT PTR
3050 RETURN
3497 :
3498 MACHINE CODE DATA
3499 :
3500 DATA 173 , 255 , 55 , 73 , 255 , 32 , 112 , 219
, 73 , 255 , 141 , 255 , 55  , 96
```



R%, it is used to discover the state of each bit of R%. C% is the character poked to the screen display, it is set to either a space, or an inverse space according to requirements. You will notice that two of the arrow characters, which are not included in the standard character sets, had to be concocted from others. If you inspect the disassembly of the machine code routine, you will notice that the parameters are passed in an inverted form, ie., every 1 has been replaced by a 0, and vice versa, this was done in order to simplify the BASIC logic, the values passed to the keyscan routine are correct.

The keyboard is in essence a set of switches that permit connections between eight lines representing the columns, and eight that represent the rows. By applying a voltage to one of the columns, and searching for a corresponding voltage on the rows, then a closed switch can be detected. For example, if the space bar is down, the voltage applied to column 7 can be detected on row 4, see FIG 2. So far so good, but we have a problem, when you ran PROGRAM 1 you may have noticed that sometimes keys were indicated as having been pressed when you had not touched them, why? Look at FIG 3, on a simple matrix like this it is possible for switches to mask each other. If 'C', 'T', and 'M' are pressed together then 'O' is detected as well, go on, try it! What happens is this, when a voltage is applied to column 4, the closed 'M' switch passes it onto row 4, then the closed 'C' passes it onto column 2, and finally the closed 'T' passes it onto row 6, where it is detected by the keyscan, of course a voltage applied to column 4 and detected on row 6 means that '0' has been pressed, doesn't it?! This phenomenon will not often cause problems, but it has to be borne in mind when selecting control keys for games etc., it would not do, for instance, if hyperspace were selected each time up, left, and the fire keys were pressed simultaneously. As a spin off, next time you run an unprotected BASIC programme, press 'N', 'M', and the space bar all together, hey presto!, RUN STOP!

| | |
|---|---|
| PL%(array) | List of player addresses. |
| TC% | Top left corner of screen. |
| D% | Parameter address (as programme 1). |
| P1% | Player 1 position |
| P2% | Player 2 position. |
| S1 | Player 1 score |
| S2 | Player 2 score. |
| T | Flag, 0, or 'crash position'. |
| SE% | Parameter passed to machine code. |
| MOVE% | Player position update value. |
| ST% | Random position of new star. |
| R% | Value returned from machine code. |
| T1% | Contents of new player 1 position. |
| T2% | Contents of new player 2 position. |
| S1% | Player 1 list start pointer. |
| S2% | Player 2 list start pointer. |
| E1% | Player 1 list end pointer. |
| E2% | Player 2 list end pointer. |

Note that PL% (array) is implemented as a circular list.




Let's have a practical example, PROGRAM 2 is a two player game that requires the detection of eight keys, several of which might be in use at any time. Each player has to have controls for up, down, left, and right, and in addition we must permit diagonal movement. To simplify the input routine, it is desirable that all of one players controls should be on one column. It so happens that if we use column 1 for player 1, and column 5 for player 2, then there are suitably placed keys for both players on rows 1, 2, 4, and 5. The machine code in PROGRAM 2 is identical to that in programme 1, and the keyscans are made in the subroutine at line 500. This time, rather than looking at the whole keyboard by means of eight column scans, only two scans are made, and the rest of the keyboard is ignored, however the interrupt vectors are untouched, RUN STOP still works.

The programme is structured for clarity rather than speed, but despite this the game is quite playable.

# SPRITE IDEAS

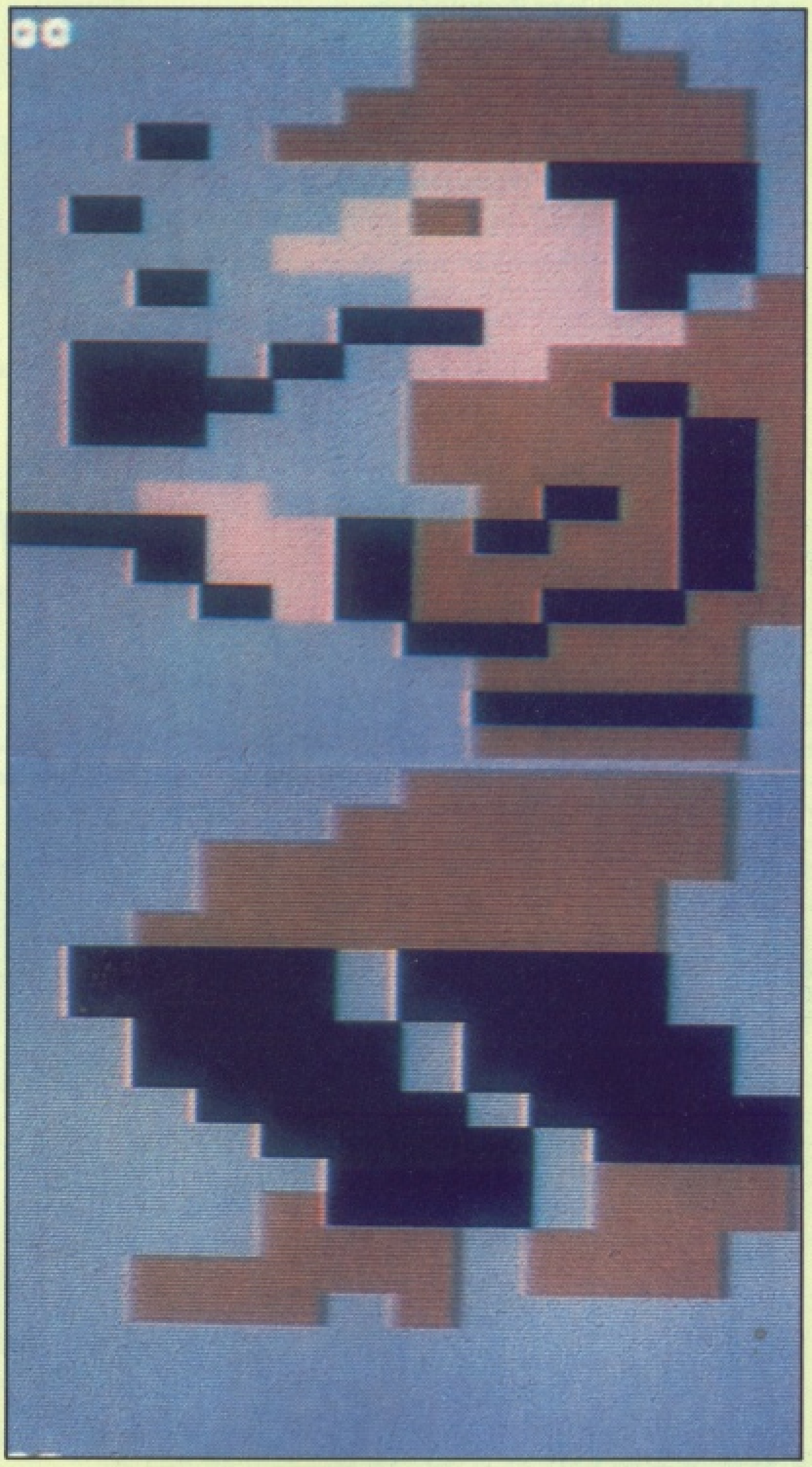
When you are designing a game one of the longest jobs is designing the sprites. If you are good at art then fine, if not your next monster will probably end up looking like a square box with legs.

Now, Your Commodore comes to the rescue once again with Sprite Ideas. If you have designed any sprites for games and you don't mind other people seeing your masterworks then why not send them into us. Each month we will be offering £10 for the best entries.

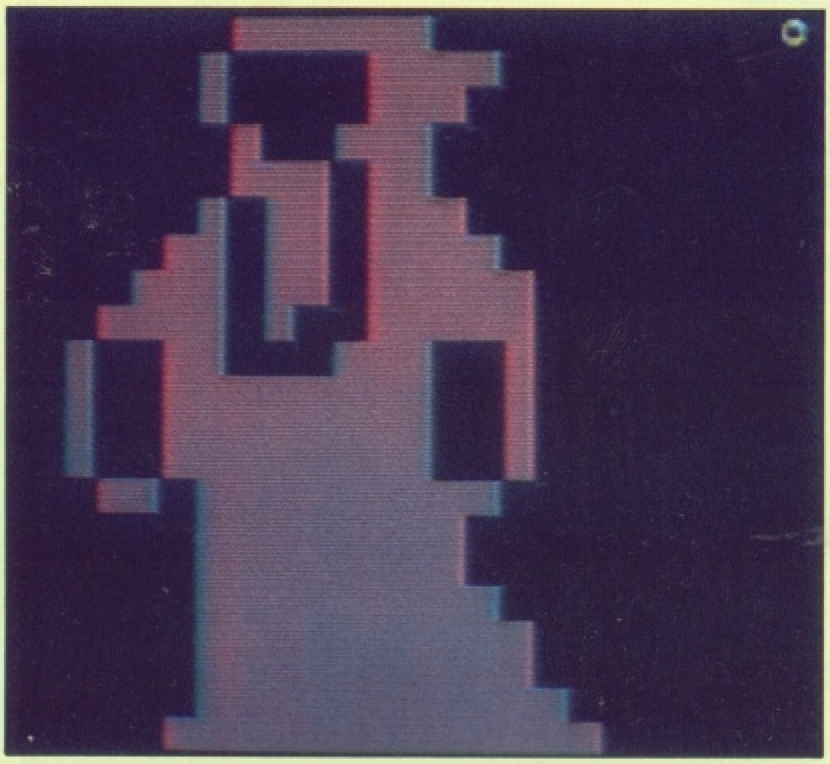
Your sprites can be anything at all (within reason), if you've designed a series of animated characters then send in the lot. We'd love to have a look at them.

So, next time you are after an Ogre to put in your new game, have a look in this section of the magazine and you may find just what you are looking for.

```
HOLMES - HEAD          DATA0,10,128,0,10,160,0,42
                       DATA168,12,170,168,0,5,252,48
STUART JAMES           DATA25,124,0,85,124,12,5,114
BIRMINGHAM             DATA0,61,90,60,197,170,63,10
                       DATA186,60,10,174,0,10,174,5
                       DATA2,238,253,123,174,13,122,174
                       DATA3,122,250,0,15,168,0,2
                       DATA168,0,3,252,0,2,168
```

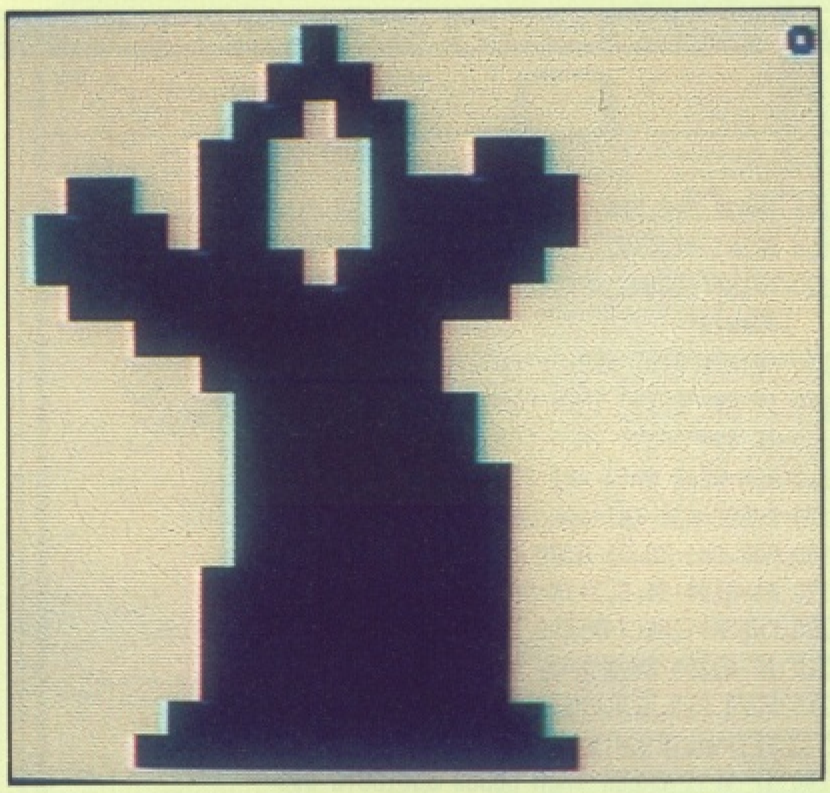


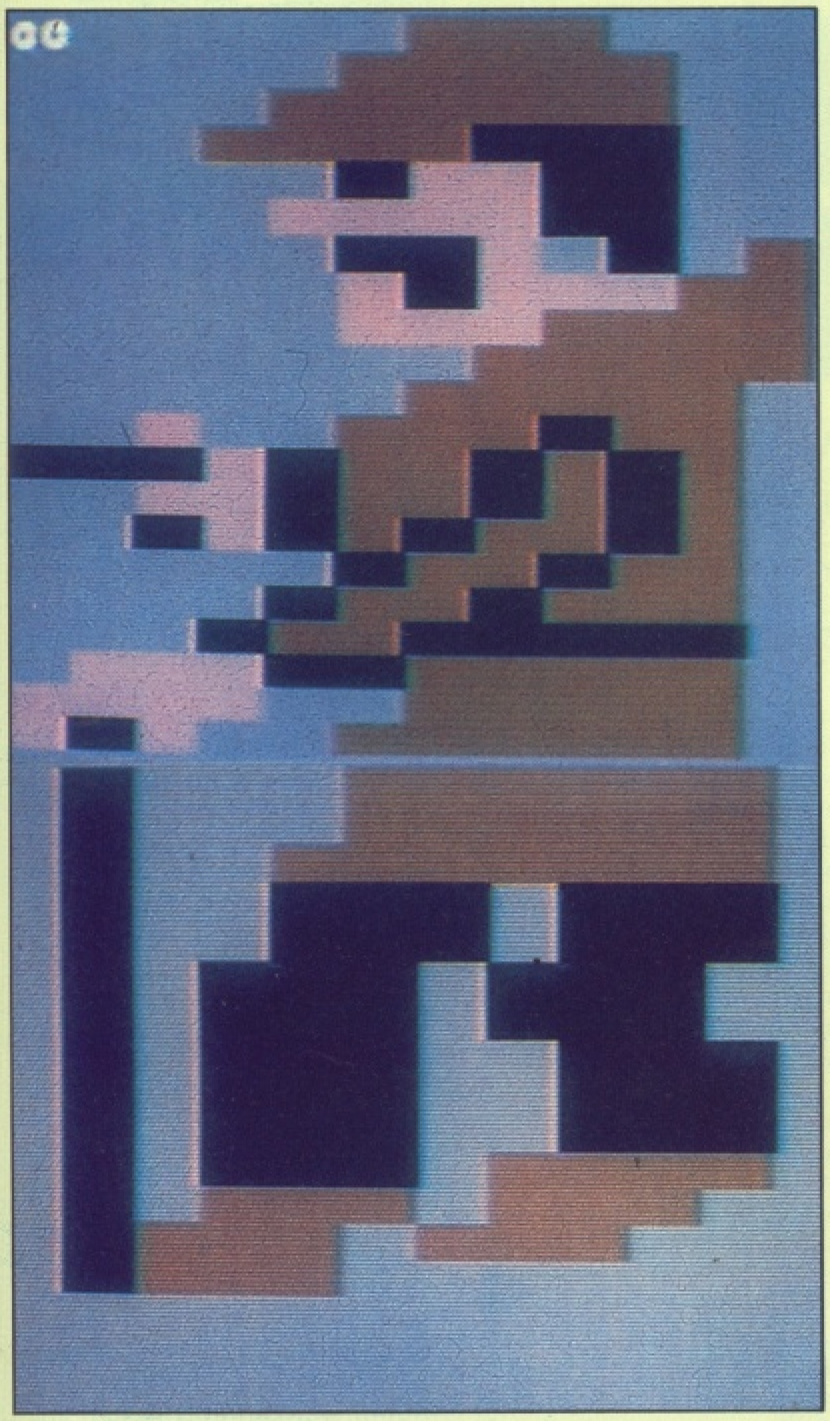
```
WIZARD                 DATA1,248,0,2,30,0,2,28
                       DATA0,1,56,0,1,216,0,2
LEE GOODMAN            DATA220,0,6,222,0,14,223,0
BERKHAMPSTED           DATA30,159,0,38,57,0,39,249
                       DATA0,39,249,0,39,249,0,27
                       DATA254,0,3,252,0,3,252,0
                       DATA3,254,0,3,255,0,3,255
                       DATA0,3,255,128,7,255,192
```



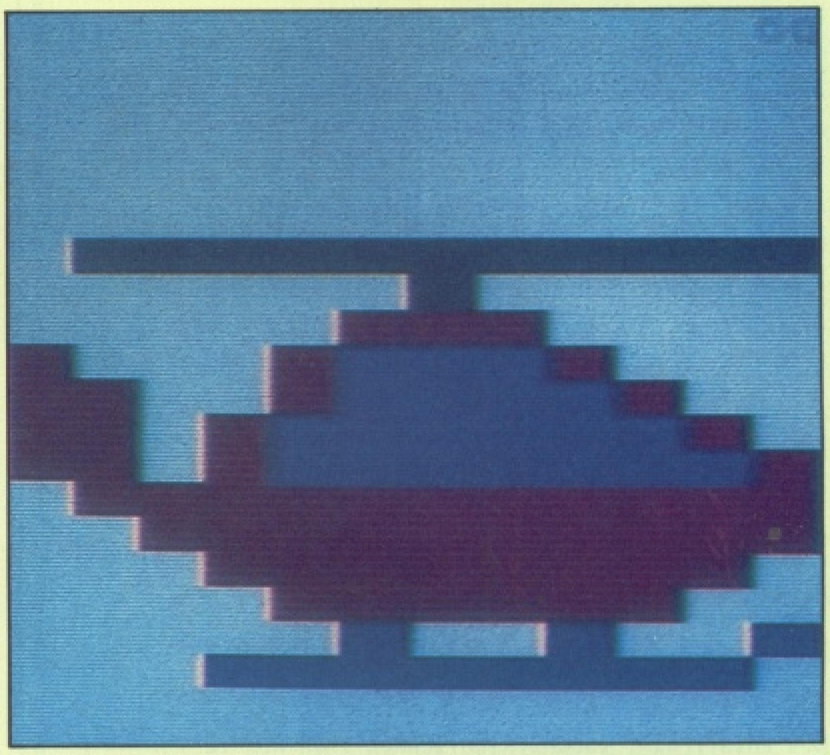
```
HOLMES.LEGS            DATA0,10,168,0,42,168,2,170
                       DATA168,2,170,160,10,170,160,63
STUART JAMES           DATA207,240,63,207,240,15,243,252
BIRMINGHAM             DATA15,243,252,3,252,255,0,255
                       DATA63,0,63,42,0,191,42,0
                       DATA168,168,10,168,168,10,136,0
                       DATA0,0,0,0,0,0,0,0
                       DATA0,0,0,0,0,0,0
```

```
WATSON-HEAD        DATA0,10,128,0,42,160,0,170
                   DATA160,2,171,240.0,53.240,0
STUART JAMES       DATA85.240.0.61.50,0,29,90
BIRMINGHAM         DATA0,21,170,0,2,170,0,10
                   DATA168.4,42,232,253.235.194,5
                   DATA235,184,13,238,184,0,58,232
                   DATA0,235,168,3,175,252.21,250
                   DATA168,85.10,168.116,42.168
```





```
GHOST              DATA0,64,0,0,224,0,1,176
                   DATA0,3,19,0.51,31.128.123
LEE GOODMAN        DATA31,128,127.191.0,63,254,0
BERKHAMPSTED       DATA15,248,0.3,248,0.1.252
                   DATA0,1,252.0,1,254,0,1
                   DATA254.0,1.254,0.3.254,0
                   DATA3.254.0,3.254,0,3,254
                   DATA0,7,255,0,15,255,128
```



```
WATSON -  LEGS     DATA48.42,168.48,42,168,48,170
                   DATA168,48,252,252,48,252,252,51
STUART JAMES       DATA243.240.51.243,240,51,240,252
BIRMINGHAM         DATA51,240.252,51.240,252,51,242
                   DATA168,50.162,160,58.138,128.58
                   DATA128,0,0,0,0,0,0,0
                   DATA0,0,0,0,0,0.0,0
                   DATA0,0,0,0,0,0,0
```

```
CHOPPER                    DATA0,0,42,170,170,0,8,0
                           DATA0,21,0,64,127,64,80,127
D DERRICK                  DATA208,81,255,244,81,255,253,21
CHEDDAR                    DATA85,85,5,85,85,1,85,84
DATA0,0,0,0,0,0,0,0        DATA0,85,80,0,48,195,3,255
DATA0,0,0,0,0,0,0,0        DATA252,0,0,0,0,0,0
```

```
100 INPUT "[CLEAR,DOWN2]START ADDRESS ";ADR
110 PRINT "[DOWN2]PLEASE ENTER ALL DIGITS."
120 PRINT"[DOWN2]SPACES WILL BE ENTERED AUTOMATICALLY
   [DOWN2]"
130 PRINT:PRINT ADR;":";:GOSUB 260
140 IF S$="[F1]" THEN GOSUB 330:GOTO 100
150 IF S$="[F3]" THEN GOSUB 460:GOTO 100
160 REM READ DATA AND STORE
170 CHECK=ADR-INT(ADR/256)*256
180 FOR C=1 TO 36 STEP 3
190 N$=MID$(S$,C,3):N=VAL(N$)
200 CHECK=(CHECK+N)AND 255
210 IF N>255 THEN N=0
220 POKE ADR,N:ADR=ADR+1:NEXT C
230 VFY=VAL(RIGHT$(S$,3))
240 IF VFY<>CHECK THEN GOSUB 530:GOTO 130
250 GOTO 130
260 S$="":FOR C=1 TO 13:FOR L=1 TO 3
270 GET K$:IF K$=""GOTO 270
280 IF K$="[F1]" THEN C=13:L=3
290 IF K$="[F3]" THEN C=13:L=3
300 S$=S$+K$
310 PRINT K$;:NEXT L:PRINT" ";:NEXT C:RETURN
320 REM **** SAVE YOUR FILE ****
330 PRINT"[CLEAR,DOWN2,RIGHT2]SAVE FILE[DOWN2]"
340 INPUT"FILE NAME :";F$
350 IF F$="" OR LEN(F$)>15 OR F$="Q" THEN RETURN
360 INPUT"[DOWN3,RVSON]T[RVSOFF]APE OR[SPC,RVSON]D[RVSOFF]
   ISK : D[LEFT3]";D$
370 D=1:IF D$="D" THEN D=8
380 INPUT"[DOWN2]START ADDRESS IN DECIMAL :";S
390 INPUT"[DOWN]END ADDRESS IN DECIMAL[SPC4]:";E
400 TS$=F$:TO=PEEK(53)+256*PEEK(54)-LEN(TS$)
   :POKE 782,TO/256
410 POKE 781,TO-PEEK(782)*256:POKE 780,LEN(TS$)
420 SYS 65469
430 POKE 780,1:POKE 781,D:POKE 782,0:SYS 65466
440 POKE 254,S/256:POKE 253,S-PEEK(254)*256:POKE 780,253
450 POKE 782,(E+1)/256:POKE 781,(E+1)-PEEK(782)*256
   :SYS 65496:RETURN
460 REM **** LOAD DATA ****
470 INPUT"[CLEAR,DOWN2,RIGHT2]FILE NAME :";F$
480 IF F$="" OR LEN(Q$)>15 THEN RETURN
490 INPUT"[DOWN3,RVSON]T[RVSOFF]APE OR[SPC,RVSON]D[RVSOFF]
   ISK : D[LEFT3]";D$
500 IF D$<>"T"AND D$<>"D"THEN RETURN
510 D=1:IF D$="D"THEN D=8
520 LOAD F$,D,1:RETURN
530 PRINT:PRINT"[DOWN2,RIGHT]ERROR":ADR=ADR-12
540 POKE 54296,15:POKE 54277,10
550 POKE 54278,100:POKE 54273,45:POKE 54272,00
   :POKE 54276,17
560 FOR X=1 TO 250:NEXT
570 POKE 54276,0:POKE 54277,0:POKE 54278,0
580 RETURN
```

# EASY ENTRY EASY ENTRY

**We make life easier for you with our machine code entry program.**

THE WORST THING ABOUT Machine Code programming is entering thousands of numbers and then finding that the program will not work. There is nothing else that you can do apart from go through all of the listing trying to locate that mistyped character which prevents the program from working correctly.

Now there's an easier way to enter your machine code programs. With the Your Commodore machine code entry program, each line of numbers is checked as soon as you press return. If you have made a mistake you will be asked to re-type the last line. Another added bonus is that you can save what you have entered at any time to tape or disk and carry on where you left off next time you come to your computer.

## Using the Loader

Before you type in any machine code program you must have typed in the machine code entry program and have it saved onto tape or disk. When you want to enter any of the machine code programs that are printed out in the form used by this program you must LOAD it into your computer. When you RUN the program you will be asked for the start address of the program. The start address is the first number in any machine code listing that appears before the colon (e.g. 49152:). You simply type in this number and press return.

All that you have to do from then on is type in all the numbers on a line. Do not type any spaces and do not type return, the program will do all of that for you. If you have made a mistake on any line the computer will ask you to type the line again. Once the line is entered correctly the computer will automatically prompt you for the next line of data.

## Saving and Loading

You can save your data to tape or disk at any time by simply entering the F1 key as the first character on any line. You will then be asked for the start and end address of the save. The start address is the first number in the listing as already mentioned. The end address is the number of the last line plus 11. Don't forget to add 11 or the last line entered will not be saved.

To load back a program that you have saved you simply have to enter the F3 key as the first item on a line. You will then be asked for the name of the program.

# DO YOU need your head examined?



**Dave Crisp takes a look at a handy utility that will allow you to re-align your disk drive.**

## How's Your Disk Drive?

APART FROM THE FACT THAT IT'S slow, does it also fail to load some of your software?

Well, the problem could be head alignment. That is, the part that moves over the disk in your drive. Because of the two types of metal used in the head mechanism high temperatures cause the metals to expand at different rates. The result of this is a sloppy head. The answer to this is to let the drive cool down and hopefully things will be OK again.

The other cause of mis-alignment is more serious. Have you noticed with some protected software the drive makes a hammering noise like a machine gun? This noise is the mechanism being 'bumped' against the stop. This repeated hammering will eventually knock the head out of position.

Would other computer owners put up with a situation where software can damage the hardware?

If you are suffering from a badly mis-aligned head then this software from Evesham Micros may be the answer.

## No Special Equipment

As far as I know this is the first disk of its type. For the first time it is possible to check and remedy head alignment without equipment such as oscilloscopes.

The disk itself has had its tracks recorded 'off-line' so that the program can interpret what it reads into a measurement.

When I received the copy of the software I was relieved as my heads were so mis-aligned that it was getting to a point where I was finding it difficult to get a directory of a disk let alone save programs with any degree of confidence.

## The Tests

There are two main tests:

**1 Speed and Clamping test**

The speed at which the disk rotates in the drive is very important and the 1541 exam shows quickly whether this could be the first of your problems.

The software takes 10 measurements of your drives speed. This is then converted into an average speed. This average speed should be within -3 to +3 of 300 rpm (1%). If this is OK the difference between the fastest sample and the slowest sample is noted and if this is greater than 0.6 then this would result in a failure.

**2 Radial Head Alignment**

The program reads what data it can from off-track disk and shows the result on a chart.

The chart is made up of a display of asterisks which shows at a glance how far out your drive head is. The chart will also show whether the mis-alignment is to the inside or outside of the track.

In the manual provided there are 14 read-outs showing results and a remedy so whatever result you get you should be able to find a chart which looks something like the one you obtain.

There is a third test which checks the position of the TRACK 1 STOP.

This is a metal casting which stops the head moving further back than track one. It is possible for this casting to become misplaced which obviously results in the head mis-aligning with that track.

The program does a stop check and shows on the chart whether you need to re-align the stop.

A printout of a chart is shown below.

## Doing The Work

If after doing the test you decide that some work on your drive is required the manual will take you through the procedure step by step. There is not enough space here to allow me to go through the procedure but you would need confidence in your ability to poke around with a screwdriver without damaging anything. If in doubt find somebody who feels a little more confident.

## Silence The Gun

You will also find in the manual a small envelope containing two soft metal springs. These replace the standard head stop. This is a simple job and one that is worth doing.

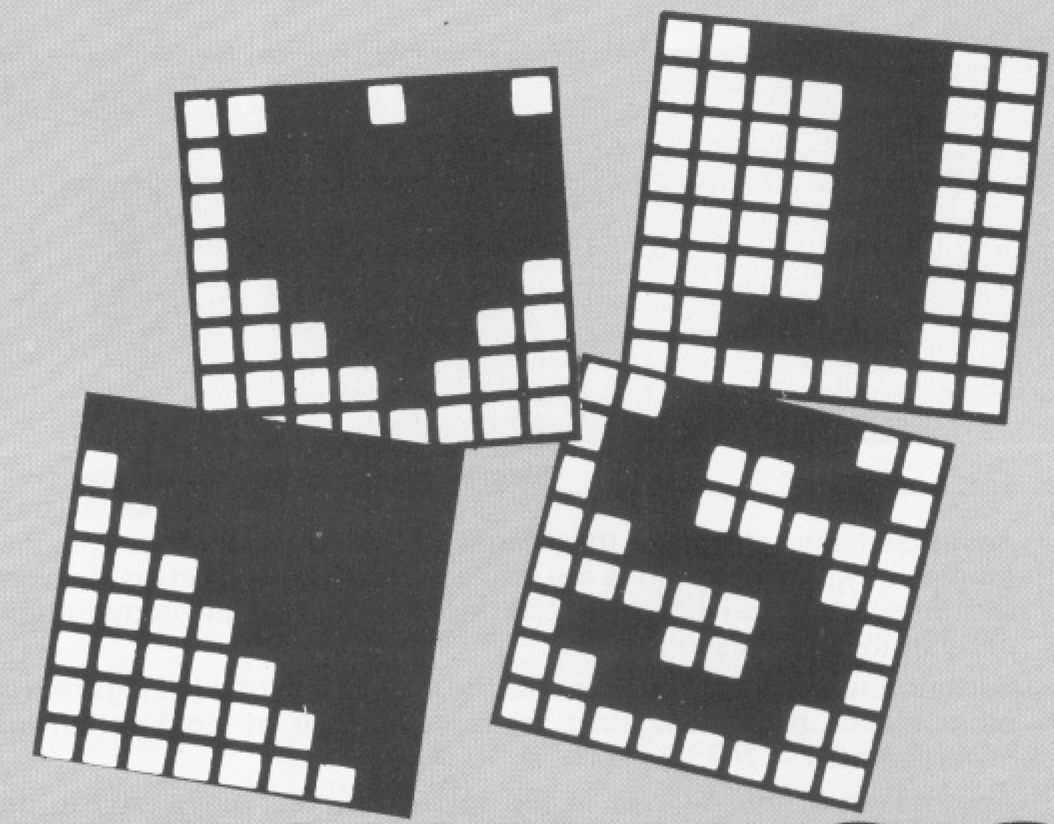
This does not stop the 'hammering' of the drive but it does make the hammering very quiet and less damaging. After fitting the 'quiet stop' run the test again to ensure that it is in the correct position.

## Conclusion

This is a good buy if you envisage problems. Of course if you head is so far out of line already you will not be able to load the diagnostics anyway. It is a useful thing to have and one which most Commodore owners would find useful at some time in their drive's life.

At £39.95 it is not cheap but could pay for itself. It is available from: Evesham Micros, Telephone 0386 41989 or 021-458564.

## Listings will be much easier to enter with our new system.



# LISTINGS

COMMODORE LISTINGS ARE RATHER well known for the horrible little black blobs that always abound. Unfortunately the graphics characters which are used to represent graphic and control characters do not reproduce very well and they are also difficult to find on the Commodore keyboard.

In future all control and graphics commands will be replaced by a mnemonic within square brackets. This mnemonic is not typed out as printed in the magazine but rather the corresponding key or keys on the keyboard are pressed. For example [RIGHT] means press the cursor right key, you do not type in [RIGHT]. All of the keywords, what keys to press and how they are shown on the screen are shown below.

Any character that is accessed by pressing shift and a letter will be printed as [Sletter].

[SA] shift and A
[S+] shift and +

Any character that is accessed by pressing the Commodore key and a letter will be printed as [Cletter]

[CA] Commodore and A
[C+] Commodore and +
[C1] Commodore and 1

If any characters are repeated the mnemonic will be followed by a number. This number is how many times you should enter the character. Any number of spaces over one will also be represented in this form

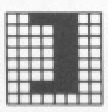
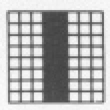
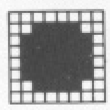
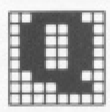
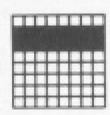
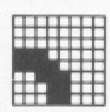
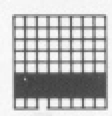
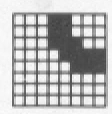
[RIGHT10] press cursor right 10 times
[C+10] press Commodore and + 10 times
[SPC10] Press the space bar 10 times

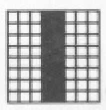
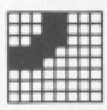
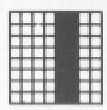
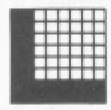
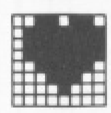
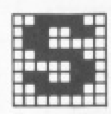
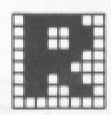
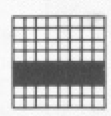
Any other characters should be easily recognisable for example CTRL-N means press CTRL and N and LEFT-ARROW means press the left arrow.

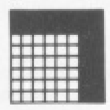
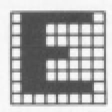
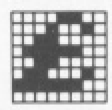
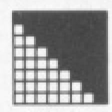
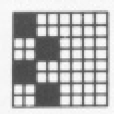
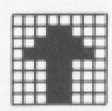
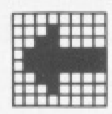
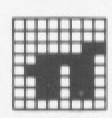
Any number of mnemonics can be enclosed in brackets for example

[SA10,SPC10,SA10]

means type 10 shift A's 10 spaces and another 10 shift A's.

| Mnemonic | Symbol | what to press |
|---|---|---|
| [RIGHT] |  | left/right |
| [LEFT] |  | shift left/right |
| [UP] |  | Shift & up /down |
| [DOWN] |  | up/down |
| [F1] |  | f1 |
| [F2] |  | shift & f1 |
| [F3] |  | f3 |
| [F4] |  | shift & f3 |

| Mnemonic | Symbol | what to press |
|---|---|---|
| [F5] |  | f5 |
| [F6] |  | shift & f5 |
| [F7] |  | f7 |
| [F8] |  | shift & f7 |
| [CLEAR] |  | shift & CLR /HOME |
| [HOME] |  | CLR/HOME |
| [RVSON] |  | CTRL & 9 |
| [RVSOFF] |  | CTRL & 0 |

| Mnemonic | Symbol | what to press |
|---|---|---|
| [BLACK] |  | CTRL & 1 |
| [WHITE] |  | CTRL & 2 |
| [RED] |  | CTRL & 3 |
| [CYAN] |  | CTRL & 4 |
| [PURPLE] |  | CTRL & 5 |
| [GREEN] |  | CTRL & 6 |
| [BLUE] |  | CTRL & 7 |
| [YELLOW] |  | CTRL & 8 |

# YOUR COMMODORE

**Lineage:** 40p per word.
**Semi display:** £9.00 per single column centimetre
Ring for information on series bookings/discounts.

All advertisements in this section must be prepaid.
Advertisements are accepted subject to the terms and conditions printed on the advertisement rate card (available on request).

**01-437 0699**
**EXT. 291**

**Send your requirements to:**
**STUART TAYLOR**
**ASP LTD, 1 GOLDEN SQUARE, LONDON W1**

## WARNING NOTICE

Advertisements placed in this magazine are to be in strict compliance with our standard conditions (copies of which conditions are available on request) and on the clear understanding that the advertiser warrants that his advertisement(s) does not infringe any copyright or condition of sale of any interested party in the advertised product.

Further, the advertiser indemnifies the proprietors of this magazine in respect of costs, damages, or any other claims brought against them as a result of legal action arising from the publication of the advertisement.

Any breach of these terms or the said conditions may result in prosecution of the advertiser by the proprietors.

## SOFTWARE

### COMMODORE 64 DATABASE — DATAFILE

Powerful options include: print all/selective records, delete, amend, save, load, copy, alphanumeric sort, user defined fields, memory check, total numeric fields, multi field interrelational, global and between limits searches, etc. Fully menu driven with comprehensive, easy to use instructions.
CBM 64 (compiled) cassette £7.45
CBM 64 (compiled) disk £10.45

### COMMODORE 64 WORDPRO — TEXTFILE

Powerful word processor, facilities include — large text buffer, auto centre, variable tab, insert, amend, delete, copy, move text, word count, right justify, paging, join files, multiple file handling, print all/part text, etc. Fully driven with comprehensive, easy to use instructions. Great value.
CBM 64 (compiled) cassette £8.25
CBM 64 (compiled) disk £10.75
All programs with full documentation and guarantee and can be used with all direct connect CBM 64 printers including 1520 print/plotter. Also most centronics (with sutiable hardware interface). Price includes fast delivery by first class post. Cheques/POs to:

**A & C SOFTWARE**
**Dept. 4, 75 Oakthorpe Gardens, Tividale, West Midlands B69 2LF**
**Telephone: 021-557 8596 (7-9pm)**

### DO YOU READ ME?

Speed reading course for children aged 10+ on CBM 64. Cassette or disk (1541) £10. Ch/POs or SAE for details to:-

**S. HARCOURT,**
**17, The Moorings,**
**North Lancing,**
**Sussex BN15 0PP**

**DISK AID**
Menu driven utility inc file recovery, directory reshuffle, fast format etc. Saves time and hassle with the 1541. **£8.95** (inc. P&P)
**BARIM SOFTWARE,**
**4, Wellington Road,**
**Ashford, Middx TW15 3RJ.**

**SPANISH, ARABIC,** Portugese, Italian word processors for CBM 64 £89 each. inc. P&P. vat extra. Dealers welcome. DT Promotions, Unit-F, Arndale Centre, London SW18 4TD. Tel: 01 870 5711. Telex 8951182 Gecoms.

**SENSATIONAL POSTAL** software bargains for Amstrad, Atari, BBC, Commodore (all formats), Electron, Spectrum. For free list send a SAE or phone Teletex Cleckheaton 0274 875299. 28 Brooklyn Court, Bradford Road, Cleckheaton, W. Yorks.

### AFFORDABLE SOFTWARE

| | |
|---|---|
| Chuckie Egg (c) | 6.75 |
| Winter Games (c) | 7.95 |
| Red Moon (c) | 5.75 |
| Snooker (c) | 6.95 |
| Goonies (c) | 7.95 |
| Solo Flight (c,d) | 11.95 |

**All prices include postage and packing. Send a S.A.E. for price list. To order any of the above titles send a cheque or postal order payable to:-**
**J + J SOFT SALES (MAIL ORDER)**
**DEPT CU, 17 VERNON ROAD, UCKFIELD, SUSSEX, TN22 5DX**

### SOUTH EAST KENT SOFTWARE

**Dept Y/C P.O. Box 152 Orpington Kent**
We stock for Spectrum, Amstrad, Commodore. Send S.A.E. for full list and prices. Please state which. Postage/ Packing 50p, overseas £1.00.

**COMMODORE 64 GAMES (TAPE)**

| | RRP | OUR PRICE |
|---|---|---|
| Super Huey | £8.95 | £6.45 |
| Ind. Jones | £9.95 | £6.50 |
| Dambusters | £9.95 | £6.90 |
| Spy vs Spy II | £9.95 | £6.50 |
| Skyfox | £9.95 | £7.50 |
| Dropzone | £9.95 | £6.99 |
| Impossible Mission | £9.95 | £6.99 |
| Summer Games | £14.95 | £11.00 |
| Break Fever | £6.95 | £5.80 |

**BUSINESS (DISK)**

| | RRP | OUR PRICE |
|---|---|---|
| Home Accounts | £24.95 | £19.95 |
| Stock Control | £24.95 | £19.95 |
| Mail List | £19.95 | £15.50 |

**EDUCATION**
English, Maths, German, French, History
**RRP** £9.95 **OUR PRICE £7.95**
Cheques/PO's payable to: **J. Ferguson.**
Please allow 21 days delivery

### BANKER 64

If you are fed up not being able to keep track of your bank account this program may be just what you are looking for. Simple to use as it is menu driven it will hold full details of Credits, Debits, Standing orders and Direct Debits and will automatically place them in date order. It will give you statements to screen or printer including the 1520 plotter printer. Records can be stored on both disk and tape for complete back-up. Entry to any account is by name and Security code which can be changed as necessary. This m/c program is very good value at **£6.25 disk, £4.00 tape.**

### RAMFILE

Keep track of your Software, Records, Stamp Collection, Addresses etc. with this easy to use menu driven filling system. Entry titles are user defineable. Powerful options include search and amend routines. Print routines allows printing of entry title, selected entry or complete file to any printer including the 1520 plotter. Files can be stored on both disk and tape.
This m/c program is very good value at **£6.25 disk, £4.00 tape.**

***Special Offer* — Both programs £10.Disk Both programs £7.00 Tape**
All prices include VAT P&P Cheques & P.O. to
**RJP ELECTRONICS**
**27 Denham Way, Camber, Rye, East Sussex TN31 7XP Tel: (0797) 225549**

**NEW COMMODORE 64** software on disc and tape. I have: Karate Champion, Winter Games, Silent Service, Last V8, Dan Dare; and many new games. Write for more information: N. Arntz, Osdorperweg 565 Amsterdam, Holland. Tel: 020 190669.

## SOFTWARE GAMES

**BLOCKBREAKER — C64** wordgame includes; Challenge Games, Gold runs, money, prizes. £5.75 Cheque/PO to Drive, 92 Robin Hood Lane, Birmingham B28 0JX.

## GRAPHICS

**PIXEL-PAL**
For Commodore 64
- Plan your Graphics
- Simple to use
- 50 A3 sheets
- Programming Guide

**COMPUTER BITS**
**35 Windermere Drive, Biggleswade, Beds.**

## TERMS

### CLASSIFIED ADVERTISING TERMS & CONDITIONS

**Our terms for new advertisers (semi-display and lineage) are strictly pro-forma payments until satisfactory reference can be taken up (excluding recognised advertising agencies). Cheques and PO's should be crossed and made payable to:**
**ARGUS SPECIALIST PUBICATIONS LTD.,**
**and send together with the advertisement to:**
**THE CLASSIFIED DEPT., L/H, NO: 1 GOLDEN SQUARE, LONDON, W1R 3AB.**
**There are no reimbursements for cancellations. Advertisements arriving too late for a particlular issue will be inserted in the folowing issue unless accompanied by instructions to the contrary.**
**All advertising sales are subject to Government regulations concerning VAT. Advertisers are responsible for complying with the various legal requirements in force eg: The Trade Description Act, Sex Discrimination Act & The Business Advertisements (Disclosure) Order 1977.**
*FULL TERMS & CONDITIONS OF ADVERTISING AVAILABLE ON REQUEST.*

## FOR HIRE



## HARDWARE





## REPAIRS





## BOOKS





## FOR SALE





## FOR HIRE



## FOR SALE

## ACCESSORIES

**PERIPHERAL BARGAINS**

Quickshot II Joystick **£8.75**
Gunshot II Joystick **£8.75**
C-16 adaptor (for above) **£3.95**
Reset switch (VIC 20/ CBM 64) **£3.95**

Will reset all m-code programs. Price includes Basic recovery program.

Copier CP1 Digital interface. Copies all Turbos, m/code, etc. IC controlled. No software needed. Just type load. If it loads, CP1 will copy it. Requires second datasette. 12 months guarantee £9.50.

Free P&P UK, Europe add £1.50, rest add £2.50.
Send Ch/PO to:

**DATALINE**
**246 Horbury Rd.,**
**Wakefield WF2 8RE.**

## FOR HIRE

**COMMODORE 64** cassette library. Hire latest games and educational software, only one price £1.50 per cassette 7 days hire. Details S.A.E. please: C.C.L., 19 Darrell Way, Abingdon, OX14 1HQ.

## FOR HIRE

**BUY OR SELL** your unwanted software, books, and hardware. We put you directly in contact with each other in your area. Details: Second Hand Computer Sales Agency. 38, Bowling Alley Walks, Dorchester, Dorset. Tel (0305) 64849 (24 hrs).

## SOFTWARE APPLICATIONS

### HOME ACCOUNTS

Put your house in order! Probably home computing's best use! Comprehensive coverage of bank accounts. Credit cards, HP in-built accuracy check. Reocrds all transactions. Projects cash-flow for any period ahead. Available for C.16 CBM64 or VIC-20. £8.45 or *free* details from:

**Discus Software**
**Freepost,**
**Beach Approach**
**Brixham TQ5 8BR**
**Tel: 08045 55532**

## FOR SALE

**CBM64 SOFTWARE DISCOUNTS**
Spy v Spy 2 £7.30, Kennedy Approach £10.95, Sabre Wolf £7.30, Underwurlde £7.30, Hacker £7.20, Rocky Horror Show £6.65. Cheque/PO to:
**Software Unlimited, PO Box 65,**
**8 Thurland St., Nottingham NG1.**
**Tel: (0602) 470897.**
Full list with order.

**BRITISH BATTLESHIPS** 1892-1960. CBM 64. Four preferential/educational cassette volumes £2.85 each or information from C. Lewis, 84 Station Road, Wraysbury Berks TW19 5NH.




**DON'T LEAVE IT TO CHANCE!**
**GIVE YOUR BUSINESS A BOOST**
**BY ADVERTISING IN**
**YOUR COMMODORE**
**PHONE STUART TAYLOR ON 01-437 0699**
**FOR DETAILS**

## YOUR COMMODORE — CLASSIFIED ADVERTISEMENT — ORDER FORM

Advertise nationally in these columns for only 40p per word (min charge 15 words). Simply print your message in the coupon and send with your cheque or postal order made payable to **Argus Specialist Publications Ltd** to:

**STUART TAYLOR**
**ASP LTD, 1 GOLDEN SQUARE,**
**LONDON W1**

**Tel: 01-437 0699**

**Name**.......................................

**Address**....................................

..................................................

..................................................

**Tel no (day)**..............................

**ADVERTISERS PLEASE ADD 15% VAT**

Please place my advert in **YOUR COMMODORE** for ........ issues.

Please indicate no. of insertions required.


A NEW CLASSIC
FROM PAUL WOAKES
AUTHOR OF ENCOUNTER!
YOU CHOOSE
THE ACTION IN
MERCENARY
Mercenary – a unique combination of flight simulation, adventure and arcade fun. You choose the action.
There is complete freedom of movement in a truly three-dimensional vector-graphic environment. Graphics of exceptional speed create a very realistic experience.
You'll never play the same game twice. Random elements attempt to seal your fate. Your interaction is crucial.
Mercenary presents an absorbing challenge that you will accept again and again.
MERCENARY
ESCAPE FROM TARG
Fly above the planet Targ in a variety of craft. Investigate the multitude of structures on the planet surface in your quest for essential supplies. Search the mysterious subterranean complex in pursuit of your fortune.
Become embroiled in the continuing conflict between the Palyars and Mechanoids. Accept their paid assignments or work independently to gain wealth.
You only have one true friend – the ever-reliable Benson, an almost human PC, who guides you in your travels.
Pursue your objective as you will : To succeed you have to escape from Targ.
NOVAGEN
INITIAL RELEASE
FOR COMMODORE 64 ATARI 48K 800 XL 130 HE
CASSETTE £9.95
DISK £12.95
NOVAGEN SOFTWARE LTD: SALES 142 ALCESTER ROAD BIRMINGHAM B13 8HS

# BUSICALC 3

## - the sophisticated spreadsheet !

Easy to learn, easy to use - something that can't be said of many business programs. But it's true of all the programs in the BUSICALC series.

BUSICALC 3 can handle all sorts of jobs - budgets, expenditure analysis, stock lists, price lists, and product costing are just a few of the possibilities. Three-dimensional formulae automatically access data stored on disk, so that you can easily pull together information from several different sheets and summarise or manipulate it.

It's simple to transfer data to other programs such as Easy Script. And you can use virtually any printer with BUSICALC 3, whether dot matrix or daisy wheel, Commodore or non-Commodore.

For the CBM 64 and PET/CBM 4000 & 8000 series.

Available through dealers or from:
**Supersoft, Winchester House, Canning Road, Harrow HA3 7SJ**

**Phone 01-861 1166 for more details and a free catalogue.**

